

```

DDDDDDDDDDDDDD RRRRRRRRRRRR IIIIIIIIII VVV VVV EEEEEEEEEEEEEEE RRRRRRRRRRRR
DDDDDDDDDDDDDD RRRRRRRRRRRR IIIIIIIIII VVV VVV EEEEEEEEEEEEEEE RRRRRRRRRRRR
DDDDDDDDDDDDDD RRRRRRRRRRRR IIIIIIIIII VVV VVV EEEEEEEEEEEEEEE RRRRRRRRRRRR
DDD DDD DDD RRR RRR RRR VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR RRR VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR RRR VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR RRR VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR RRR VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR RRR VVV VVV EEE RRR RRR
DDD DDD DDD RRRRRRRRRRRR IIIIIIIIII VVV VVV EEEEEEEEEEEEE RRRRRRRRRRRR
DDD DDD DDD RRRRRRRRRRRR IIIIIIIIII VVV VVV EEEEEEEEEEEEE RRRRRRRRRRRR
DDD DDD DDD RRRRRRRRRRRR IIIIIIIIII VVV VVV EEEEEEEEEEEEE RRRRRRRRRRRR
DDD DDD DDD RRR RRR IIIIIIIIII VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR IIIIIIIIII VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR IIIIIIIIII VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR RRR IIIIIIIIII VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR RRR IIIIIIIIII VVV VVV EEE RRR RRR
DDD DDD DDD RRR RRR RRR IIIIIIIIII VVV VVV EEE RRR RRR
DDDDDDDDDDDDDD RRR RRR RRR IIIIIIIIII VVV VVV EEEEEEEEEEEEE RRR RRR
DDDDDDDDDDDDDD RRR RRR RRR IIIIIIIIII VVV VVV EEEEEEEEEEEEE RRR RRR
DDDDDDDDDDDDDD RRR RRR RRR IIIIIIIIII VVV VVV EEEEEEEEEEEEE RRR RRR

```

[illegible]

```

XX      XX      AAAAAA      DDDDDDDD      RRRRRRRR      IIIIII      VV      VV      EEEEEEEEEEE      RRRRRRRR
XX      XX      AAAAAA      DDDDDDDD      RRRRRRRR      RRRRRRRR      VV      VV      EEEEEEEEEEE      RRRRRRRR
XX      XX      AA      AA      DD      DD      RR      RR      II      II      EE      RR      RR
XX      XX      AA      AA      DD      DD      RR      RR      II      II      EE      RR      RR
XX      XX      AA      AA      DD      DD      RR      RR      II      II      EE      RR      RR
XX      XX      AA      AA      DD      DD      RR      RR      II      II      EE      RR      RR
XX      XX      AA      AA      DD      DD      RRRRRRRR      II      II      VV      VV      EEEEEEEEEEE      RRRRRRRR
XX      XX      AA      AA      DD      DD      RRRRRRRR      II      II      VV      VV      EEEEEEEEEEE      RRRRRRRR
XX      XX      AAAAAAAAAA      DD      DD      RR      RR      II      II      VV      VV      EE      RR      RR
XX      XX      AAAAAAAAAA      DD      DD      RR      RR      II      II      VV      VV      EE      RR      RR
XX      XX      AA      AA      DD      DD      RR      RR      II      II      VV      VV      EE      RR      RR
XX      XX      AA      AA      DD      DD      RR      RR      II      II      VV      VV      EE      RR      RR
XX      XX      AA      AA      DDDDDDDD      RR      RR      IIIIII      VV      VV      EEEEEEEEEEE      RR      RR
XX      XX      AA      AA      DDDDDDDD      RR      RR      IIIIII      VV      VV      EEEEEEEEEEE      RR      RR

```

```

LL               IIIIII               SSSSSSSS
LL               IIIIII               SSSSSSSS
LL               II                   SS
LL               II                   SS
LL               II                   SS
LL               II                   SS
LL               II                   SSSSSS
LL               II                   SSSSSS
LL               II                   SS
LL               II                   SS
LL               II                   SS
LL               II                   SS
LLLLLLLLLLLLLL  IIIIII               SSSSSSSS
LLLLLLLLLLLLLL  IIIIII               SSSSSSSS

```


(2)	81	External and local symbol definitions
(3)	225	Device Driver Tables
(4)	285	XA_CONTROL_INIT, Controller initialization
(5)	332	XA_READ_WRITE, FDT for device data transfers
(6)	410	XA_SETMODE, Set Mode, Set characteristics FDT
(7)	477	XA_START, Start I/O routines
(10)	931	DR11-W DEVICE TIME-OUT
(11)	974	XA_INTERRUPT, Interrupt service routine for DR11-W
(12)	1047	XA_REGISTER - Handle DR11-W CSR transfers
(13)	1092	XA_CANCEL, Cancel I/O routine
(14)	1147	DEC_ATTNAST, Deliver ATTN AST's
(15)	1200	XA_REGDUMP - DR11-W register dump routine
(16)	1256	XA_DEV_RESET - Device reset DR11-W

```
0000 1 .TITLE XADRIVER - VAX/VMS DR11-W DRIVER
0000 2 .IDENT 'V04-001'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 *
0000 8 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 * ALL RIGHTS RESERVED.
0000 11 *
0000 12 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 * TRANSFERRED.
0000 18 *
0000 19 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 * CORPORATION.
0000 22 *
0000 23 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 *
0000 26 *****
0000 27
0000 28 ++
0000 29
0000 30 FACILITY:
0000 31
0000 32 VAX/VMS Executive, I/O Drivers
0000 33
0000 34 ABSTRACT:
0000 35
0000 36 This module contains the DR11-W driver:
0000 37
0000 38 Tables for loading and dispatching
0000 39 Controller initialization routine
0000 40 FDT routine
0000 41 The start I/O routine
0000 42 The interrupt service routine
0000 43 Device specific Cancel I/O
0000 44 Error logging register dump routine
0000 45
0000 46 ENVIRONMENT:
0000 47
0000 48 Kernal Mode, Non-paged
0000 49
0000 50 AUTHOR:
0000 51
0000 52 C. A. Sameulson 10-JAN-79
0000 53
0000 54
0000 55 MODIFIED BY:
0000 56
0000 57 V04-001 JLV0395 Jake VanNoy 6-SEP-1984
```


0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :--

Add AVL bit to DEVCHAR.

V03-006 TMK0001 Todd M. Katz 07-Dec-1983
Fix a broken branch.

V03-005 JLV0304 Jake VanNoy 24-AUG-1983
Several bug fixes. All word writes to XA_CSR now have
ATTN set so as to prevent lost interrupts. Attention
AST list is synchronized at device IPL in DEL_ATTNAST.
Correct status is returned on a set mode ast that
is returns through EXESFINISHIO. REQCOM's are always
done at FIPL. Signed division that prevented full size
transfers has been fixed.

V03-004 KDM0059 Kathleen D. Morse 14-Jul-1983
Change time-wait loops to use new TIMEDWAIT macro.
Add \$DEVDEF.

V03-003 KDM0002 Kathleen D. Morse 28-Jun-1982
Added \$DYNDEF, \$DCDEF, and \$SSDEF.


```
0000 81      .SBTTL External and local symbol definitions
0000 82
0000 83
0000 84      ; External symbols
0000 85
0000 86      $ACBDEF      ; AST control block
0000 87      $CRBDEF      ; Channel request block
0000 88      $DCDEF       ; Device types
0000 89      $DDBDEF      ; Device data block
0000 90      $DEVDEF      ; Device characteristics
0000 91      $DPTDEF      ; Driver prolog table
0000 92      $DYNDEF      ; Dynamic data structure types
0000 93      $EMBDEF      ; EMB offsets
0000 94      $IOBDEF      ; Interrupt data block
0000 95      $IODEF       ; I/O function codes
0000 96      $IPLDEF      ; Hardware IPL definitions
0000 97      $IRPDEF      ; I/O request packet
0000 98      $PRDEF       ; Internal processor registers
0000 99      $PRIDEF      ; Scheduler priority increments
0000 100     $SSDEF       ; System status codes
0000 101     $UCBDEF      ; Unit control block
0000 102     $VECDEF      ; Interrupt vector block
0000 103     $XADEF       ; Define device specific characteristics
0000 104
0000 105     ; Local symbols
0000 106
0000 107     ; Argument list (AP) offsets for device-dependent QIO parameters
0000 108
00000000 0000 109 P1      = 0      ; First QIO parameter
00000004 0000 110 P2      = 4      ; Second QIO parameter
00000008 0000 111 P3      = 8      ; Third QIO parameter
0000000C 0000 112 P4      = 12     ; Fourth QIO parameter
00000010 0000 113 P5      = 16     ; Fifth QIO parameter
00000014 0000 114 P6      = 20     ; Sixth QIO parameter
0000 115
0000 116     ; Other constants
0000 117
0000000A 0000 118 XA_DEF_TIMEOUT = 10      ; 10 second default device timeout
0000FFFF 0000 119 XA_DEF_BUFSIZ  = 65535   ; Default buffer size
00000001 0000 120 XA_RESET_DELAY = <<2+9>/10> ; Delay N microseconds after RESET
0000 121      ; (rounded up to 10 microsec intervals)
0000 122
0000 123     ; DR11-W definitions that follow the standard UCB fields
0000 124     ; *** N O T E *** ORDER OF THESE UCB FIELDS IS ASSUMED
0000 125
0000 126     $DEFINI UCB
000000A0 0000 127     .=UCBSL DPC+4
000000A4 00A0 128 $DEF UCBSL_XA_ATTN      ; Attention AST listhead
000000A4 00A0 129     .BLKL 1
000000A6 00A4 130 $DEF UCBSW_XA_CSRTMP      ; Temporary storage of CSR image
000000A6 00A4 131     .BLKW 1
000000A8 00A6 132 $DEF UCBSW_XA_BARTMP      ; Temporary storage of BAR image
000000A8 00A6 133     .BLKW 1
000000AA 00A8 134 $DEF UCBSW_XA_CSR      ; Saved CSR on interrupt
000000AA 00A8 135     .BLKW 1
000000AC 00AA 136 $DEF UCBSW_XA_EIR      ; Saved EIR on interrupt
000000AC 00AA 137     .BLKW 1
```



```
000000AE 00AC 138 $DEF UCBSW_XA_IDR ; Saved IDR on interrupt
000000AE 00AC 139 .BLKW 1
000000B0 00AE 140 $DEF UCBSW_XA_BAR ; Saved BAR register on interrupt
000000B0 00AE 141 .BLKW 1
000000B2 00B0 142 $DEF UCBSW_XA_WCR ; Saved WCR register on interrupt
000000B2 00B0 143 .BLKW 1
000000B4 00B2 144 $DEF UCBSW_XA_ERROR ; Saved device status flag
000000B4 00B2 145 .BLKW 1
000000B8 00B4 146 $DEF UCBSL_XA_DPR ; Data Path Register contents
000000B8 00B4 147 .BLKL 1
000000BC 00B8 148 $DEF UCBSL_XA_FMPR ; Final Map Register contents
000000BC 00B8 149 .BLKL 1
000000C0 00BC 150 $DEF UCBSL_XA_PMPR ; Previous Map Register contents
000000C0 00BC 151 .BLKL 1
000000C2 00C0 152 $DEF UCBSW_XA_DPRN ; Saved Datapath Register Number
000000C2 00C0 153 .BLKW 1 ; And Datapath Parity error flag
00C2 154
00C2 155 ; Bit positions for device-dependent status field in UCB
00C2 156
00C2 157 $VIELD UCB,0,<- ; UCB device specific bit definitions
00C2 158 <ATTNAST,,M>,- ; ATTN AST requested
00C2 159 <UNEXPT,,M>,- ; Unexpected interrupt received
00C2 160 >
000000C2 00C2 161 UCBSK_SIZE=
00C2 162 $DEFEND UCB
0000 163
0000 164 ; Device register offsets from CSR address
0000 165
0000 166 $DEFINI XA ; Start of DR11-W definitions
0000 167 $DEF XA_WCR ; Word count
00000002 0000 168 .BLKW 1
0000 169 $DEF XA_BAR ; Buffer address
00000004 0002 170 .BLKW 1
0000 171 $DEF XA_CSR ; Control/status
0004 172
0004 173
0004 174 ; Bit positions for device control/status register
0004 175
0004 176 $EQU LST XASK,,0,1,<- ; Define CSR FNCT bit values
0004 177 <FNCT1,2>-
0004 178 <FNCT2,4>-
0004 179 <FNCT3,8>-
0004 180 <STATUSA,2048>- ; Define CSR STATUS bit values
0004 181 <STATUSB,1024>-
0004 182 <STATUSC,512>-
0004 183 >
0004 184
0004 185 $VIELD XA_CSR,0,<- ; Control/status register
0004 186 <GO,,M>,- ; Start device
0004 187 <FNCT,3,M>,- ; CSR FNCT bits
0004 188 <XBA,2,M>,- ; Extended address bits
0004 189 <IE,,M>,- ; Enable interrupts
0004 190 <RDY,,M>,- ; Device ready for command
0004 191 <CYCLE,,M>,- ; Starts slave transmit
0004 192 <STATUS,3,M>,- ; CSR STATUS bits
0004 193 <MAINT,,M>,- ; Maintenance bit
0004 194 <ATTN,,M>,- ; Status from other processor
```



```

0004 195          <NEX,,M>,-          ; Nonexistent memory flag
0004 196          <ERROR,,M>,-        ; Error or external interrupt
0004 197          >
0004 198 $DEF     XA_EIR                ; Error information register
0004 199
0004 200 ; Bit positions for error information register
0004 201
0004 202          $VIELD XA_EIR,0,<-    ; Error information register
0004 203          <REGFLG,,M>,-        ; Flags whether EIR or CSR is accessed
0004 204          <SPARE,,M>,-        ; Unused - spare
0004 205          <BURST,,M>,-        ; Burst mode transfer occurred
0004 206          <DLT,,M>,-          ; Time-out for successive burst xfer
0004 207          <PAR,,M>,-          ; Parity error during DAT1/P
0004 208          <ACLO,,M>,-        ; Power fail on this processor
0004 209          <MULTI,,M>,-        ; Multi-cycle request error
0004 210          <ATTN,,M>,-        ; ATTN - same as in CSR
0004 211          <NEX,,M>,-          ; NEX - same as in CSR
0004 212          <ERROR,,M>,-        ; ERROR - same as in CSR
0004 213          >
00000006 0004 214          .BLKW 1
0006 215
0006 216 $DEF     XA_IDR                ; Input Data Buffer register
0006 217 $DEF     XA_ODR                ; Output Data Buffer register
00000008 0006 218          .BLKW 1
0008 219
0008 220          $DEFEND XA            ; End of DR11-W definitions
0000 221
0000 222
0000 223

```



```
0000 225      .SBTTL Device Driver Tables
0000 226
0000 227 ; Driver prologue table
0000 228
0000 229      DPTAB      -      ; DPT-creation macro
0000 230      END=XA END,-      ; End of driver label
0000 231      ADAPTER=UBA,-    ; Adapter type
0000 232      FLAGS=DPT$M_SVP,- ; Allocate system page table
0000 233      UCBSIZE=UCB$K_SIZE,- ; UCB size
0000 234      NAME=XADRIVER    ; Driver name
0038 235      DPT_STORE INIT  ; Start of load
0038 236      ; initialization table
0038 237      DPT_STORE UCB,UCB$B_FIPL,B,8 ; Device fork IPL
003C 238      DPT_STORE UCB,UCB$B_DIPL,B,22 ; Device interrupt IPL
0040 239      DPT_STORE UCB,UCB$L_DEVCHAR,L,<- ; Device characteristics
0040 240      DEV$M_AVL!,-      ; Available
0040 241      DEV$M_RTM!,-      ; Real Time device
0040 242      DEV$M_ELG!,-      ; Error Logging enabled
0040 243      DEV$M_IDV!,-      ; input device
0040 244      DEV$M_ODV>        ; output device
0047 245      DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$ REALTIME ; Device class
004B 246      DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$ DR11W ; Device Type
004F 247      DPT_STORE UCB,UCB$W_DEVBUFFSIZ,W,- ; Default buffer size
004F 248      XA DEF BUFFSIZ
0054 249      DPT_STORE REINIT  ; Start of reload
0054 250      ; initialization table
0054 251      DPT_STORE DDB,DDB$L_DDT,D,XA$DDT ; Address of DDT
0059 252      DPT_STORE CRB,CRB$L_INTD+4,D,- ; Address of interrupt
0059 253      XA INTERRUPT      ; service routine
005E 254      DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,- ; Address of controller
005E 255      D,XA CONTROL_INIT ; initialization routine
0063 256      DPT_STORE END      ; End of initialization
0000 257      ; tables
0000 258
0000 259 ; Driver dispatch table
0000 260
0000 261      DDTAB      -      ; DDT-creation macro
0000 262      DEVNAM=XA,-          ; Name of device
0000 263      START=XA START,-    ; Start I/O routine
0000 264      FUNCTB=XA FUNCTABLE,- ; FDT address
0000 265      CANCEL=XA_CANCEL,-   ; Cancel I/O routine
0000 266      REGDMP=XA_REGDUMP,-   ; Register dump routine
0000 267      DIAGBF=<<T3*4>+<<3+5+1>*4>>,- ; Diagnostic buffer size
0000 268      ERLGBF=<<13*4>+<1*4>+<EMB$L_DV_REGSABV>> ; Error log buffer size
0038 269
0038 270 ;
0038 271 ; Function dispatch table
0038 272 ;
0038 273 XA_FUNCTABLE: ; FDT for driver
0038 274      FUNCTAB      -      ; Valid I/O functions
0038 275      <READPBLK,READLBLK,READVBLK,WRITEPBLK,WRITELBLK,WRITEVBLK,-
0038 276      SETMODE,SETCHAR,SENSEMODE,SENSECHAR>
0040 277      FUNCTAB      ; No buffered functions
0048 278      FUNCTAB XA READ WRITE,- ; Device-specific FDT
0048 279      <READPBLK,READLBLK,READVBLK,WRITEPBLK,WRITELBLK,WRITEVBLK>
0054 280      FUNCTAB +EX$READ,<READPBLK,READLBLK,READVBLK>
0060 281      FUNCTAB +EX$WRITE,<WRITEPBLK,WRITELBLK,WRITEVBLK>
```


XADRIVER
V04-001

- VAX/VMS DR11-W DRIVER
Device Driver Tables

E 16

16-SEP-1984 00:14:45 VAX/VMS Macro V04-00
6-SEP-1984 16:32:52 [DRIVER.SRC]XADRIVER.MAR;2

Page 7
(3)

006C 282
0078 283

FUNCTAB XA SETMODE,<SETMODE,SETCHAR>
FUNCTAB +EXE\$SENSEMODE,<SENSEMODE,SENSECHAR>


```
0084 285 .SBTTL XA_CONTROL_INIT, Controller initialization
0084 286
0084 287 :++
0084 288 : XA_CONTROL_INIT, Called when driver is loaded, system is booted, or
0084 289 : power failure recovery.
0084 290
0084 291 : Functional Description:
0084 292
0084 293 : 1) Allocates the direct data path permanently
0084 294 : 2) Assigns the controller data channel permanently
0084 295 : 3) Clears the Control and Status Register
0084 296 : 4) If power recovery, requests device time-out
0084 297
0084 298 : Inputs:
0084 299
0084 300 : R4 = address of CSR
0084 301 : R5 = address of IDB
0084 302 : R6 = address of DDB
0084 303 : R8 = address of CRB
0084 304
0084 305 : Outputs:
0084 306
0084 307 : VEC$V_PATHLOCK bit set in CRB$L_INTD+VEC$B_DATAPATH
0084 308 : UCB address placed into IDB$L_OWNER
0084 309
0084 310 :--
0084 311
0084 312
0084 313 XA_CONTROL_INIT:
0084 314
0084 315 : 50 18 A5 D0 0084 315 MOVL IDB$L_UCBLST(R5),R0 ; Address of UCB
0084 316 : 04 A5 50 D0 0088 316 MOVL R0,IDB$L_OWNER(R5) ; Make permanent controller owner
0084 317 : 64 A0 10 A8 008C 317 BISW #UCB$M_ONLINE,UCB$W_STS(R0)
0090 318 : ; Set device status "on-line"
0090 319
0090 320 : If powerfail has occurred and device was active, force device time-out.
0090 321 : The user can set his own time-out interval for each request. Time-
0090 322 : out is forced so a very long time-out period will be short circuited.
0090 323
0090 324 : 05 64 A0 05 E0 0090 324 BBS #UCB$V_POWER,UCB$W_STS(R0),10$
0095 325 : ; Branch if powerfail
0095 326 : 37 A8 80 8F 88 0095 326 BISB #VEC$M_PATHLOCK,CRB$L_INTD+VEC$B_DATAPATH(R8)
009A 327 : ; Permanently allocate direct datapath
009A 328 10$:
009A 328 BSBW XA_DEV_RESET ; Reset DR11W
009A 329 RSB ; Done
009D 330
```



```
009E 332 .SBTTL XA_READ_WRITE, FDT for device data transfers
009E 333
009E 334 :++
009E 335 : XA_READ_WRITE, FDT for READBLK,READVBLK,READPBLK,WRITEBLK,WRITEVBLK,
009E 336 : WRITEPBLK
009E 337
009E 338 : Functional description:
009E 339
009E 340 : 1) Rejects QUEUE I/O's with odd transfer count
009E 341 : 2) Rejects QUEUE I/O's for BLOCK MODE request to UBA Direct Data
009E 342 : PATH on odd byte boundary
009E 343 : 3) Stores request time-out count specified in P3 into IRP
009E 344 : 4) Stores FNCT bits specified in P4 into IRP
009E 345 : 5) Stores word to write into ODR from P5 into IRP
009E 346 : 6) Checks block mode transfers for memory modify access
009E 347
009E 348 : Inputs:
009E 349
009E 350 : R3 = Address of IRP
009E 351 : R4 = Address of PCB
009E 352 : R5 = Address of UCB
009E 353 : R6 = Address of CCB
009E 354 : R8 = Address of FDT routine
009E 355 : AP = Address of P1
009E 356 : P1 = Buffer Address
009E 357 : P2 = Buffer size in bytes
009E 358 : P3 = Request time-out period (conditional on IOSM_TIMED)
009E 359 : P4 = Value for CSR FNCT bits (conditional on IOSM_SETFNCT)
009E 360 : P5 = Value for ODR (conditional on IOSM_SETFNCT)
009E 361 : P6 = Address of Diagnostic Buffer
009E 362
009E 363 : Outputs:
009E 364
009E 365 : R0 = Error status if odd transfer count
009E 366 : IRP$M_MEDIA = Time-out count for this request
009E 367 : IRP$M_SEGVBN = FNCT bits for DR11-W CSR and ODR image
009E 368
009E 369 :--
009E 370
009E 371 XA_READ_WRITE:
009E 372
009E 373 2$: BLBC P2(AP),10$ : Branch if transfer count even
009E 374 5$: MOVZWL #SS$ BADPARAM,R0 : Set error status code
009E 375 10$: JMP G*EX$ABORTIO : Abort request
009E 376 : MOVZWL IRP$M_FUNC(R3),R1 : Fetch I/O Function code
009E 377 : MOVL P3(AP),IRP$M_MEDIA(R3) : Set request specific time-out count
009E 378 : BBS #IOSV_TIMED,R1,15$ : Branch if time-out specified
009E 379 : MOVL #XA_DEF_TIMEOUT,IRP$M_MEDIA(R3) : Else set default timeout value
009E 380 15$: BBC #IOSV_DIAGNOSTIC,R1,20$ : Branch if not maintenance request
009E 381 : EXTZV #IOSV_FCODE,#IOS$F_CODE,R1,R1 : AND out all function modifiers
009E 382 : CMPB #IOS_READPBLK,R1 : If maintenance function, must be
009E 383 : : physical I/O read or write
009E 384 : BEQL 20$
009E 385 : CMPB #IOS_WRITEPBLK,R1
009E 386 : BEQL 20$
009E 387 : MOVZWL #SS$_NOPRIV,R0 : No privilege for operation
009E 388 : BRB 5$ : Abort request
```

09 04 AC	E9	009E	372
50 14 3C	00A2	373	2\$:
00000000 GF	17 00A5	374	5\$:
51 20 A3	3C 00AB	375	10\$:
38 A3 08 AC	D0 00AF	376	
04 51 07 E0	00B4	377	
38 A3 0A D0	00B8	378	
	00BC	379	
14 51 08 E1	00BC	380	15\$:
51 51 06 00 EF	00C0	381	
51 0C 91	00C5	382	
	00C8	383	
	0A 13 00C8	384	
51 08 91	00CA	385	
	05 13 00CD	386	
50 24 3C	00CF	387	
D1 11 00D2	388		


```
50 0C AC 03 00 EF 00D4 389 20$: EXTZV #0,#3,P4(AP),R0 ; Get value for FNCT bits
    48 A3 50 01 78 00DA 390 ASHL #XA_C$R$V FNCT,R0,IRP$L_SEGVB$N(R3) ; Shift into position for CSR
    4A A3 10 AC B0 00DF 391 MOVW P5(AP),IRP$L_SEGVB$N+2(R3) ; Store ODR value for later
    00E4 392
    00E4 393 ; If this is a block mode transfer, check buffer for modify access
    00E4 394 ; whether or not the function is read or write. The DR11-W does
    00E4 395 ; not decide whether to read or write, the users device does.
    00E4 396 ; For word mode requests, return to read check or write check.
    00E4 397
    00E4 398 ; If this is a BLOCK MODE request and the UBA Direct Data Path is
    00E4 399 ; in use, check the data buffer address for word alignment. If buffer
    00E4 400 ; is not word aligned, reject the request.
    00E4 401
    0E 20 A3 06 E0 00E4 402 BBS #IOS$V_WORD,IRP$W_FUNC(R3),30$
    03 44 A5 00 E0 00E9 403 ; Branch if word mode transfer
    00EE 404 BBS #XA$V_DATAPATH,UCB$L_DEVDEPEND(R5),25$
    00EE 405 ; Branch if Buffered Data Path in use
    B1 6C E8 00EE 406 BLBS P1(AP),2$ ; DDP, branch on bad alignment
    00000000'GF 17 00F1 407 25$: JMP G^EXE$MODIFY ; Check buffer for modify access
    05 00F7 408 30$: RSB ; Return
```



```
00F8 410 .SBTTL XA_SETMODE, Set Mode, Set characteristics FDT
00F8 411
00F8 412 :++
00F8 413 : XA_SETMODE, FDT routine to process SET MODE and SET CHARACTERISTICS
00F8 414
00F8 415 : Functional description:
00F8 416
00F8 417 : If IOSM_ATTNAST modifier is set, queue attention AST for device
00F8 418 : If IOSM_DATAPATH modifier is set, queue packet.
00F8 419 : Else, finish I/O.
00F8 420
00F8 421 : Inputs:
00F8 422
00F8 423 : R3 = I/O packet address
00F8 424 : R4 = PCB address
00F8 425 : R5 = UCB address
00F8 426 : R6 = CCB address
00F8 427 : R7 = Function code
00F8 428 : AP = QIO Parameter list address
00F8 429
00F8 430 : Outputs:
00F8 431
00F8 432 : If IOSM_ATTNAST is specified, queue AST on UCB attention AST list.
00F8 433 : If IOSM_DATAPATH is specified, queue packet to driver.
00F8 434 : Else, use exec routine to update device characteristics
00F8 435
00F8 436 :--
00F8 437
00F8 438 XA_SETMODE:
50 20 A3 3C 00F8 439 MOVZWL IRPSW_FUNC(R3),R0 ; Get entire function code
2B 50 08 E1 00FC 440 BBC #IOSV_ATTNAST,R0,20$ ; Branch if not an ATTN AST
0100 441
0100 442 : Attention AST request
0100 443
0090 8F BB 0100 444 PUSHR #*M<R4,R7>
57 00A0 C5 9E 0104 445 MOVAB UCB$L_XA_ATTN(R5),R7 ; Address of ATTN AST control block list
00000000'GF 16 0109 446 JSB G*COM$SETATTNAST ; Set up attention AST
0090 8F BA 010F 447 POPR #*M<R4,R7>
2D 50 E9 0113 448 BLBC R0,50$ ; Branch if error
68 A5 01 A8 0116 449 BISW #UCB$M_ATTNAST,UCB$W_DEVSTS(R5)
011A 450 ; Flag ATTN AST expected.
03 68 A5 01 E1 011A 451 BBC #UCB$V_UNEXPT,UCB$W_DEVSTS(R5),10$
011F 452 ; Deliver AST if unsolicited interrupt
045E 30 011F 453 BSBW DEL_ATTNAST
50 01 9A 0122 454 10$: MOVZBL #SS$NORMAL,R0 ; Set status
00000000'GF 17 0125 455 JMP G*EXE$FINISHIOC ; Thats all for now (clears R1)
012B 456
012B 457 : If modifier IOSM_DATAPATH is set,
012B 458 : queue packet. The data path is changed at driver level to preserve
012B 459 : order with other requests.
012B 460
06 50 0A E0 012B 461 20$: BBS S*#IOSV_DATAPATH,R0,30$ ; If BDP modifier set, queue packet
012F 462
00000000'GF 17 012F 463 JMP G*EXE$SETCHAR ; Set device characteristics
0135 464
0135 465 : This is a request to change data path useage, queue packet
0135 466
```


XADRIVER
V04-001

- VAX/VMS DR11-W DRIVER

J 16

16-SEP-1984 00:14:45

VAX/VMS Macro V04-00

Page 12

XA_SETMODE, Set Mode, Set characteristic

6-SEP-1984 16:32:52

[DRIVER.SRC]XADRIVER.MAR;2

(6)

```

57 1A D1 0135 467 30$: CMPL #IOS_SETCHAR,R7 ; Set characteristics?
      06 12 0138 468      BNEQ 45$ ; No, must have the privilege
00000000'GF 17 013A 469      JMP G^EXE$SETMODE ; Queue packet to start I/O
      0140 470
      0140 471 ; Error, abort IO
      0140 472
50 24 3C 0140 473 45$: MOVZWL #SS$_NOPRIV,R0 ; No priv for operation
      51 D4 0143 474 50$: CLRL R1
00000000'GF 17 0145 475      JMP G^EXE$ABORTIO ; Abort IO on error
```



```
014B 477 .SBTTL XA_START, Start I/O routines
014B 478 :++
014B 479 : XA_START - Start a data transfer, set characteristics, enable ATTN AST.
014B 480 :
014B 481 : Functional Description:
014B 482 :
014B 483 : This routine has two major functions:
014B 484 :
014B 485 : 1) Start an I/O transfer. This transfer can be in either word
014B 486 : or block mode. The FNCTN bits in the DR11-W CSR are set. If
014B 487 : the transfer count is zero, the STATUS bits in the DR11-W CSR
014B 488 : are read and the request completed.
014B 489 : 2) Set Characteristics. If the function is change data path, the
014B 490 : new data path flag is set in the UCB.
014B 491 :
014B 492 : Inputs:
014B 493 :
014B 494 : R3 = Address of the I/O request packet
014B 495 : R5 = Address of the UCB
014B 496 :
014B 497 : Outputs:
014B 498 :
014B 499 : R0 = final status and number of bytes transferred
014B 500 : R1 = value of CSR STATUS bits and value of input data buffer register
014B 501 : Device errors are logged
014B 502 : Diagnostic buffer is filled
014B 503 :
014B 504 :--
014B 505 .ENABL LSB
014B 506
014B 507 XA_START:
014B 508
014B 509 : Retrieve the address of the device CSR
014B 510
014B 511 ASSUME IDB$$_CSR EQ 0
014B 512 MOVL UCB$$_CRB(R5),R4 ; Address of CRB
014B 513 MOVL @CRB$$_INTD+VEC$$_IDB(R4),R4 ; Address of CSR
014B 514
014B 515
014B 516 : Fetch the I/O function code
014B 517
014B 518 MOVZWL IRP$$_FUNC(R3),R1 ; Get entire function code
014B 519 MOVW R1,UCB$$_FUNC(R5) ; Save FUNC in UCB for Error Logging
014B 520 EXTZV #IOSV_FCODE,#IOSS_FCODE,R1,R2 ; Extract function field
014B 521
014B 522 : Dispatch on function code. If this is SET CHARACTERISTICS, we will
014B 523 : select a data path for future use.
014B 524 : If this is a transfer function, it will either be processed in word
014B 525 : or block mode.
014B 526
014B 527 CMPB #IOS$_SETCHAR,R2 ; Set characteristics?
014B 528 BNEQ 3$
014B 529
014B 530 :++
014B 531 : SET CHARACTERISTICS - Process Set Characteristics QIO function
014B 532 :
014B 533 : INPUTS:
```

54 24 A5 D0
54 2C B4 D0

52 51 20 A3 3C
009A C5 51 B0
51 06 00 EF

52 1A 91
23 12


```
0166 534 :  
0166 535 : XA_DATAPATH bit in Device Characteristics specifies which data path  
0166 536 : to use. If bit is a one, use buffered data path. If zero, use  
0166 537 : direct datapath.  
0166 538 :  
0166 539 : OUTPUTS:  
0166 540 :  
0166 541 : CRB is flagged as to which datapath to use.  
0166 542 : DEVDEPEND bits in device characteristics is updated  
0166 543 : XA_DATAPATH = 1 -> buffered data path in use  
0166 544 : XA_DATAPATH = 0 -> direct data path in use  
0166 545 :--  
0166 546 :  
40 50 24 A5 D0 0166 547 MOVL UCB$$_CRB(R5),R0 ; Get CRB address  
A5 38 A3 7D 016A 548 MOVQ IRP$$_MEDIA(R3),UCB$$_DEVCLASS(R5) ; Set device characteristics  
37 A0 80 8F 88 016F 549 BISB #VEC$$_PATHLOCK,CRB$$_INTD+VEC$$_DATAPATH(R0)  
0174 550 ; Assume direct datapath  
05 44 A5 00 E1 0174 551 BBC #XASV_DATAPATH,UCB$$_DEVDEPEND(R5),2$ ; Were we right?  
37 A0 80 8F 8A 0179 552 BICB #VEC$$_PATHLOCK,CRB$$_INTD+VEC$$_DATAPATH(R0) ; Set buffered datapath  
017E 553 2$: CLRL R1 ; Return Success  
50 51 D4 017E 554 MOVZWL #SS$$_NORMAL,R0  
01 3C 0180 555 REQCOM  
0183 556  
0189 557  
0189 558 ; If subfunction modifier for device reset is set, do one here  
0189 559  
03 51 0B E1 0189 560 3$: BBC S^#IOSV_RESET,R1,4$ ; Branch if not device reset  
046E 30 018D 561 BSBW XA_DEV_RESET ; Reset DR11-W  
0190 562  
0190 563 ; This must be a data transfer function - i.e. READ OR WRITE  
0190 564 ; Check to see if this is a zero length transfer.  
0190 565 ; If so, only set CSR FNCT bits and return STATUS from CSR  
0190 566  
7E A5 B5 0190 567 4$: TSTW UCB$$_BCNT(R5) ; Is transfer count zero?  
59 12 0193 568 BNEQ 10$ ; No, continue with data transfer  
2E 51 09 E1 0195 569 BBC S^#IOSV_SETFNCT,R1,6$ ; Set CSR FNCT specified?  
0199 570 DSBINT  
06 A4 4A A3 B0 019F 571 MOVW IRP$$_SEGVBN+2(R3),XA_ODR(R4)  
01A4 572 ; Store word in ODR  
50 04 A4 3C 01A4 573 MOVZWL XA_CSR(R4),R0  
50 800E 8F AA 01A8 574 BICW #<XA_CSR$$_FNCT!XA_CSR$$_ERROR>,R0  
50 48 A3 A8 01AD 575 BISW IRP$$_SEGVBN(R3),R0  
50 2000 8F A8 01B1 576 BISW #XA_CSR$$_ATTN,R0 ; Force ATTN on to prevent lost interrupt  
04 A4 50 B0 01B6 577 MOVW R0,XA_CSR(R4)  
05 44 A5 01 E1 01BA 578 BBC #XASV_LINK,UCB$$_DEVDEPEND(R5),5$ ; Link mode?  
04 A4 50 04 AB 01BF 579 BICW3 #XASK_FNCT2,R0,XA_CSR(R4) ; Make FNCT bit 2 a pulse  
01C4 580 5$: ENBINT  
01C4 581  
01C7 582 6$: BSBW XA_REGISTER ; Fetch DR11-W registers  
032B 30 01C7 583 BLBS R0,7$ ; If error, then log it  
06 50 E8 01CA 584 JSB G^ERL$$_DEVICERR ; Log a device error  
00000000'GF 16 01CD 585 JSB G^IOC$$_DIAGBUFILL ; Fill diagnostic buffer if specified  
00000000'GF 16 01D3 586 7$: MOVW UCB$$_XA_CSR(R5),R1 ; Return CSR and EIR in R1  
51 00A8 C5 D0 01D9 587 MOVZWL UCB$$_XA_ERROR(R5),R0 ; Return status in R0  
50 00B2 C5 3C 01DE 588 BISB #XA_CSR$$_IE,XA_CSR(R4) ; Enable device interrupts  
04 A4 40 8F 88 01E3 589 REQCOM ; Request done  
01E8 590
```



```
01EE 591
01EE 592 ; Build CSR image in R0 for later use in starting transfers
01EE 593
01EE 594 10$:
00B4 C5 50 7E A5 3C 01EE 595 MOVZWL UCBSW_BCNT(R5),R0 ; Fetch byte count
C7 01F2 596 DIVL3 #2,R0,UCBSL_XA_DPR(R5) ; Make byte count into word count
01F8 597
01F8 598 ; Set up UCBSW_CSRTMP used for loading CSR later
01F8 599
01F8 600 MOVZWL XA_CSR(R4),R0
50 50 04 A4 3C 01F8 601 BICW #^C<XA_CSR$M_FNCT>,R0
50 FFF1 8F AA 01FC 602 BICW #XA_CSR$M_IET,XA_CSR$M_ATTN,R0 ; Set Interrupt Enable and ATTN
50 2040 8F AB 0201 603 BISW S^#IOSV SETFNCT,R1,20$ ; Set FNCT bits in CSR?
07 51 09 E1 0206 604 BBC S^#IOSV SETFNCT,R1,20$ ; Yes, Clear previous FNCT bits
50 50 0E AA 020A 605 BICW #<XA_CSR$M_FNCT>,R0 ; OR in new value
50 48 A3 88 020D 606 BISB IRPSC_SEGVBN(R3),R0 ; Check for maintenance function
05 51 08 E1 0211 607 BBC S^#IOSV DIAGNOSTIC,R1,23$ ; Set maintenance bit in CSR image
50 1000 8F AB 0215 608 BISW #XA_CSR$M_MAINT,R0
021A 609 ; Is this a word mode or block mode request?
021A 610
00A4 C5 50 B0 021A 611 23$: MOVW R0,UCBSW_XA_CSRTMP(R5) ; Save CSR image in UCB
03 51 06 E1 021F 612 BBC S^#IOSV WORD,R1,BLOCK_MODE ; Check if word or block mode
013A 31 0223 613 BRW WORD_MODE ; Branch to handle word mode
```



```
0226 615 :++
0226 616 : BLOCK MODE -- Process a Block Mode (DMA) transfer request
0226 617 :
0226 618 : FUNCTIONAL DESCRIPTION:
0226 619 :
0226 620 : This routine takes the buffer address, buffer size, function code,
0226 621 : and function modifier fields from the IRP. It calculates the UNIBUS
0226 622 : address, allocates the UBA map registers, loads the DR11-W device
0226 623 : registers and starts the request.
0226 624 :--
0226 625 : Set up UBA
0226 626 : Start transfer
0226 627 :
0226 628 BLOCK_MODE:
0226 629 :
0226 630 : If IOSM_CYCLE subfunction is specified, set CYCLE bit in CSR image
0226 631 :
0226 632 BBC #IOSM_CYCLE,R1,25$ ; Set CYCLE bit in CSR?
00A4 C5 07 51 0C E1 0226 633 BISW #XA_CSRSM_CYCLE,UCBSW_XA_CSRTMP(R5) ; If yes, or into CSR image
0226 634 :
0231 635 : Allocate UBA data path and map registers
0231 636 :
0231 637 25$:
0231 638 REQDPR ; Request UBA data path
0237 639 REQMPR ; Request UBA map registers
023D 640 LOADUBA ; Load UBA map registers
0243 641 :
0243 642 : Calculate the UNIBUS transfer address for the DR11-W from the UBA
0243 643 : map register address and byte offset.
0243 644 :
0243 645 MOVZWL UCBSW_BOFF(R5),R1 ; Byte offset in first page of xfer
51 09 51 7C A5 3C 0243 646 MOVL UCBSL_CRB(R5),R2 ; Address of CRB
0247 647 INSV CRBSL_INTD+VECSW_MAPREG(R2),#9,#9,R1
024B 648 : Insert page number
0251 649 EXTZV #16,#2,R1,R2 ; Extract bits 17:16 of bus address
52 51 02 10 EF 0251 650 ASHL #XA_CSR$V_XBA,R2,R2 ; Shift extended memory bits for CSR
52 52 52 04 78 025A 651 BISW #XA_CSRSM_GO,R2 ; Set "GO" bit into CSR image
00A4 C5 52 01 AB 025D 652 BISW R2,UCBSW_XA_CSRTMP(R5) ; Set into CSR image we are building
50 00A4 C5 0101 BF AB 0262 653 BICW3 #<XA_CSRSM_GO!XA_CSRSM_CYCLE>,UCBSW_XA_CSRTMP(R5),R0
026A 654 : CSR image less "GO" and "CYCLE"
52 00A4 C5 04 AB 026A 655 BICW3 #XASK_FNCT2,UCBSW_XA_CSRTMP(R5),R2 ; CSR image less FNCT bit 2
00A6 C5 51 B0 0270 656 MOVW R1,UCBSW_XA_BARTMP(R5) ; Save BAR for error logging
0275 657 :
0275 658 : At this juncture:
0275 659 : R0 = CSR image less "GO" and "CYCLE"
0275 660 : R1 = low 16 bits of transfer bus address
0275 661 : R2 = CSR image less FNCT bit 2
0275 662 : UCBSL_XA_DPR(R5) = transfer count in words
0275 663 : UCBSW_XA_CSRTMP(R5) = CSR image to start transfer with
0275 664 :
0275 665 : Set DR11-W registers and start transfer
0275 666 : Note that read-modify-write cycles are NOT performed to the DR11-W CSR.
0275 667 : The CSR is always written directly into. This prevents inadvertently setting
0275 668 : the EIR select flag (writing bit 15) if error happens to become true.
0275 669 :
0275 670 DSBINT ; Disable interrupts (powerfail)
64 00B4 C5 AE 027B 671 MNEGW UCBSL_XA_DPR(R5),XA_WCR(R4)
```



```
02 A4 51 B0 0280 672 ; Load negative of transfer count
04 A4 50 B0 0280 673 ; Load low 16 bits of bus address
06 44 A5 01 E1 0284 674 ; Load CSR image less "GO" and "CYCLE"
04 A4 52 B0 0288 675 ; Load CSR image less "GO" and "CYCLE"
06 11 0291 676 ; Link mode?
04 A4 00A4 C5 B0 028D 677 ; Yes, load CSR image less "FNCT" bit 2
06 11 0291 677 ; Only if link mode in dev characteristics
04 A4 00A4 C5 B0 0293 678 26$: MOVW UCBSW_XA_CSRTMP(R5),XA_CSR(R4) ; Move all bits to CSR
06 11 0293 679 ; Wait for transfer complete interrupt, powerfail, or device time-out
04 A4 00A4 C5 B0 0299 680 126$: WFIKPCX XA_TIME_OUT,IRPSL_MEDIA(R3) ; Wait for interrupt
06 11 0299 681 ; Device has interrupted, FORK
04 A4 00A4 C5 B0 0299 682 IOFORK ; FORK to lower IPL
06 11 0299 683 ; Handle request completion, release UBA resources, check for errors
04 A4 00A4 C5 B0 0299 684
06 11 0299 685
04 A4 00A4 C5 B0 02A4 686
06 11 02A4 687
04 A4 00A4 C5 B0 02A4 688
06 11 02A4 689
04 A4 00A4 C5 B0 02AA 690
06 11 02AA 691
04 A4 00A4 C5 B0 02AA 692
06 11 02AA 693
04 A4 00A4 C5 B0 02AD 694
06 11 02AD 695
04 A4 00A4 C5 B0 02B1 696
06 11 02B1 697
04 A4 00A4 C5 B0 02B7 698
06 11 02B7 699
04 A4 00A4 C5 B0 02BA 700
06 11 02BA 701
04 A4 00A4 C5 B0 02BF 702
06 11 02BF 703
04 A4 00A4 C5 B0 02C3 704
06 11 02C3 705
04 A4 00A4 C5 B0 02C8 706
06 11 02C8 707
04 A4 00A4 C5 B0 02CA 708
06 11 02CA 709
04 A4 00A4 C5 B0 02CB 710
06 11 02CB 711
04 A4 00A4 C5 B0 02CE 712
06 11 02CE 713
04 A4 00A4 C5 B0 02D3 714
06 11 02D3 715
04 A4 00A4 C5 B0 02DA 716
06 11 02DA 717
04 A4 00A4 C5 B0 02E1 718
06 11 02E1 719
04 A4 00A4 C5 B0 02E6 720
06 11 02E6 721
04 A4 00A4 C5 B0 02EB 722
06 11 02EB 723
04 A4 00A4 C5 B0 02ED 724
06 11 02ED 725
04 A4 00A4 C5 B0 02F3 726
06 11 02F3 727
04 A4 00A4 C5 B0 02F7 728
06 11 02F7 729
04 A4 00A4 C5 B0 02F9 730
06 11 02F9 731
04 A4 00A4 C5 B0 02FC 732
06 11 02FC 733
04 A4 00A4 C5 B0 02FF 734
06 11 02FF 735
04 A4 00A4 C5 B0 0301 736
06 11 0301 737
04 A4 00A4 C5 B0 0307 738
06 11 0307 739
04 A4 00A4 C5 B0 030D 740
06 11 030D 741
04 A4 00A4 C5 B0 0313 742
06 11 0313 743
04 A4 00A4 C5 B0 0313 744
06 11 0313 745
04 A4 00A4 C5 B0 0313 746
06 11 0313 747
04 A4 00A4 C5 B0 0313 748
06 11 0313 749
04 A4 00A4 C5 B0 0313 750
06 11 0313 751
04 A4 00A4 C5 B0 0313 752
06 11 0313 753
04 A4 00A4 C5 B0 0313 754
06 11 0313 755
04 A4 00A4 C5 B0 0313 756
06 11 0313 757
04 A4 00A4 C5 B0 0313 758
06 11 0313 759
04 A4 00A4 C5 B0 0313 760
06 11 0313 761
04 A4 00A4 C5 B0 0313 762
06 11 0313 763
04 A4 00A4 C5 B0 0313 764
06 11 0313 765
04 A4 00A4 C5 B0 0313 766
06 11 0313 767
04 A4 00A4 C5 B0 0313 768
06 11 0313 769
04 A4 00A4 C5 B0 0313 770
06 11 0313 771
04 A4 00A4 C5 B0 0313 772
06 11 0313 773
04 A4 00A4 C5 B0 0313 774
06 11 0313 775
04 A4 00A4 C5 B0 0313 776
06 11 0313 777
04 A4 00A4 C5 B0 0313 778
06 11 0313 779
04 A4 00A4 C5 B0 0313 780
06 11 0313 781
04 A4 00A4 C5 B0 0313 782
06 11 0313 783
04 A4 00A4 C5 B0 0313 784
06 11 0313 785
04 A4 00A4 C5 B0 0313 786
06 11 0313 787
04 A4 00A4 C5 B0 0313 788
06 11 0313 789
04 A4 00A4 C5 B0 0313 790
06 11 0313 791
04 A4 00A4 C5 B0 0313 792
06 11 0313 793
04 A4 00A4 C5 B0 0313 794
06 11 0313 795
04 A4 00A4 C5 B0 0313 796
06 11 0313 797
04 A4 00A4 C5 B0 0313 798
06 11 0313 799
04 A4 00A4 C5 B0 0313 800
06 11 0313 801
04 A4 00A4 C5 B0 0313 802
06 11 0313 803
04 A4 00A4 C5 B0 0313 804
06 11 0313 805
04 A4 00A4 C5 B0 0313 806
06 11 0313 807
04 A4 00A4 C5 B0 0313 808
06 11 0313 809
04 A4 00A4 C5 B0 0313 810
06 11 0313 811
04 A4 00A4 C5 B0 0313 812
06 11 0313 813
04 A4 00A4 C5 B0 0313 814
06 11 0313 815
04 A4 00A4 C5 B0 0313 816
06 11 0313 817
04 A4 00A4 C5 B0 0313 818
06 11 0313 819
04 A4 00A4 C5 B0 0313 820
06 11 0313 821
04 A4 00A4 C5 B0 0313 822
06 11 0313 823
04 A4 00A4 C5 B0 0313 824
06 11 0313 825
04 A4 00A4 C5 B0 0313 826
06 11 0313 827
04 A4 00A4 C5 B0 0313 828
06 11 0313 829
04 A4 00A4 C5 B0 0313 830
06 11 0313 831
04 A4 00A4 C5 B0 0313 832
06 11 0313 833
04 A4 00A4 C5 B0 0313 834
06 11 0313 835
04 A4 00A4 C5 B0 0313 836
06 11 0313 837
04 A4 00A4 C5 B0 0313 838
06 11 0313 839
04 A4 00A4 C5 B0 0313 840
06 11 0313 841
04 A4 00A4 C5 B0 0313 842
06 11 0313 843
04 A4 00A4 C5 B0 0313 844
06 11 0313 845
04 A4 00A4 C5 B0 0313 846
06 11 0313 847
04 A4 00A4 C5 B0 0313 848
06 11 0313 849
04 A4 00A4 C5 B0 0313 850
06 11 0313 851
04 A4 00A4 C5 B0 0313 852
06 11 0313 853
04 A4 00A4 C5 B0 0313 854
06 11 0313 855
04 A4 00A4 C5 B0 0313 856
06 11 0313 857
04 A4 00A4 C5 B0 0313 858
06 11 0313 859
04 A4 00A4 C5 B0 0313 860
06 11 0313 861
04 A4 00A4 C5 B0 0313 862
06 11 0313 863
04 A4 00A4 C5 B0 0313 864
06 11 0313 865
04 A4 00A4 C5 B0 0313 866
06 11 0313 867
04 A4 00A4 C5 B0 0313 868
06 11 0313 869
04 A4 00A4 C5 B0 0313 870
06 11 0313 871
04 A4 00A4 C5 B0 0313 872
06 11 0313 873
04 A4 00A4 C5 B0 0313 874
06 11 0313 875
04 A4 00A4 C5 B0 0313 876
06 11 0313 877
04 A4 00A4 C5 B0 0313 878
06 11 0313 879
04 A4 00A4 C5 B0 0313 880
06 11 0313 881
04 A4 00A4 C5 B0 0313 882
06 11 0313 883
04 A4 00A4 C5 B0 0313 884
06 11 0313 885
04 A4 00A4 C5 B0 0313 886
06 11 0313 887
04 A4 00A4 C5 B0 0313 888
06 11 0313 889
04 A4 00A4 C5 B0 0313 890
06 11 0313 891
04 A4 00A4 C5 B0 0313 892
06 11 0313 893
04 A4 00A4 C5 B0 0313 894
06 11 0313 895
04 A4 00A4 C5 B0 0313 896
06 11 0313 897
04 A4 00A4 C5 B0 0313 898
06 11 0313 899
04 A4 00A4 C5 B0 0313 900
06 11 0313 901
04 A4 00A4 C5 B0 0313 902
06 11 0313 903
04 A4 00A4 C5 B0 0313 904
06 11 0313 905
04 A4 00A4 C5 B0 0313 906
06 11 0313 907
04 A4 00A4 C5 B0 0313 908
06 11 0313 909
04 A4 00A4 C5 B0 0313 910
06 11 0313 911
04 A4 00A4 C5 B0 0313 912
06 11 0313 913
04 A4 00A4 C5 B0 0313 914
06 11 0313 915
04 A4 00A4 C5 B0 0313 916
06 11 0313 917
04 A4 00A4 C5 B0 0313 918
06 11 0313 919
04 A4 00A4 C5 B0 0313 920
06 11 0313 921
04 A4 00A4 C5 B0 0313 922
06 11 0313 923
04 A4 00A4 C5 B0 0313 924
06 11 0313 925
04 A4 00A4 C5 B0 0313 926
06 11 0313 927
04 A4 00A4 C5 B0 0313 928
06 11 0313 929
04 A4 00A4 C5 B0 0313 930
06 11 0313 931
04 A4 00A4 C5 B0 0313 932
06 11 0313 933
04 A4 00A4 C5 B0 0313 934
06 11 0313 935
04 A4 00A4 C5 B0 0313 936
06 11 0313 937
04 A4 00A4 C5 B0 0313 938
06 11 0313 939
04 A4 00A4 C5 B0 0313 940
06 11 0313 941
04 A4 00A4 C5 B0 0313 942
06 11 0313 943
04 A4 00A4 C5 B0 0313 944
06 11 0313 945
04 A4 00A4 C5 B0 0313 946
06 11 0313 947
04 A4 00A4 C5 B0 0313 948
06 11 0313 949
04 A4 00A4 C5 B0 0313 950
06 11 0313 951
04 A4 00A4 C5 B0 0313 952
06 11 0313 953
04 A4 00A4 C5 B0 0313 954
06 11 0313 955
04 A4 00A4 C5 B0 0313 956
06 11 0313 957
04 A4 00A4 C5 B0 0313 958
06 11 0313 959
04 A4 00A4 C5 B0 0313 960
06 11 0313 961
04 A4 00A4 C5 B0 0313 962
06 11 0313 963
04 A4 00A4 C5 B0 0313 964
06 11 0313 965
04 A4 00A4 C5 B0 0313 966
06 11 0313 967
04 A4 00A4 C5 B0 0313 968
06 11 0313 969
04 A4 00A4 C5 B0 0313 970
06 11 0313 971
04 A4 00A4 C5 B0 0313 972
06 11 0313 973
04 A4 00A4 C5 B0 0313 974
06 11 0313 975
04 A4 00A4 C5 B0 0313 976
06 11 0313 977
04 A4 00A4 C5 B0 0313 978
06 11 0313 979
04 A4 00A4 C5 B0 0313 980
06 11 0313 981
04 A4 00A4 C5 B0 0313 982
06 11 0313 983
04 A4 00A4 C5 B0 0313 984
06 11 0313 985
04 A4 00A4 C5 B0 0313 986
06 11 0313 987
04 A4 00A4 C5 B0 0313 988
06 11 0313 989
04 A4 00A4 C5 B0 0313 990
06 11 0313 991
04 A4 00A4 C5 B0 0313 992
06 11 0313 993
04 A4 00A4 C5 B0 0313 994
06 11 0313 995
04 A4 00A4 C5 B0 0313 996
06 11 0313 997
04 A4 00A4 C5 B0 0313 998
06 11 0313 999
04 A4 00A4 C5 B0 0313 1000
06 11 0313 1001
04 A4 00A4 C5 B0 0313 1002
06 11 0313 1003
04 A4 00A4 C5 B0 0313 1004
06 11 0313 1005
04 A4 00A4 C5 B0 0313 1006
06 11 0313 1007
04 A4 00A4 C5 B0 0313 1008
06 11 0313 1009
04 A4 00A4 C5 B0 0313 1010
06 11 0313 1011
04 A4 00A4 C5 B0 0313 1012
06 11 0313 1013
04 A4 00A4 C5 B0 0313 1014
06 11 0313 1015
04 A4 00A4 C5 B0 0313 1016
06 11 0313 1017
04 A4 00A4 C5 B0 0313 1018
06 11 0313 1019
04 A4 00A4 C5 B0 0313 1020
06 11 0313 1021
04 A4 00A4 C5 B0 0313 1022
06 11 0313 1023
04 A4 00A4 C5 B0 0313 1024
06 11 0313 1025
04 A4 00A4 C5 B0 0313 1026
06 11 0313 1027
04 A4 00A4 C5 B0 0313 1028
06 11 0313 1029
04 A4 00A4 C5 B0 0313 1030
06 11 0313 1031
04 A4 00A4 C5 B0 0313 1032
06 11 0313 1033
04 A4 00A4 C5 B0 0313 1034
06 11 0313 1035
04 A4 00A4 C5 B0 0313 1036
06 11 0313 1037
04 A4 00A4 C5 B0 0313 1038
06 11 0313 1039
04 A4 00A4 C5 B0 0313 1040
06 11 0313 1041
04 A4 00A4 C5 B0 0313 1042
06 11 0313 1043
04 A4 00A4 C5 B0 0313 1044
06 11 0313 1045
04 A4 00A4 C5 B0 0313 1046
06 11 0313 1047
04 A4 00A4 C5 B0 0313 1048
06 11 0313 1049
04 A4 00A4 C5 B0 0313 1050
06 11 0313 1051
04 A4 00A4 C5 B0 0313 1052
06 11 0313 1053
04 A4 00A4 C5 B0 0313 1054
06 11 0313 1055
04 A4 00A4 C5 B0 0313 1056
06 11 0313 1057
04 A4 00A4 C5 B0 0313 1058
06 11 0313 1059
04 A4 00A4 C5 B0 0313 1060
06 11 0313 1061
04 A4 00A4 C5 B0 0313 1062
06 11 0313 1063
04 A4 00A4 C5 B0 0313 1064
06 11 0313 1065
04 A4 00A4 C5 B0 0313 1066
06 11 0313 1067
04 A4 00A4 C5 B0 0313 1068
06 11 0313 1069
04 A4 00A4 C5 B0 0313 1070
06 11 0313 1071
04 A4 00A4 C5 B0 0313 1072
06 11 0313 1073
04 A4 00A4 C5 B0 0313 1074
06 11 0313 1075
04 A4 00A4 C5 B0 0313 1076
06 11 0313 1077
04 A4 00A4 C5 B0 0313 1078
06 11 0313 1079
04 A4 00A4 C5 B0 0313 1080
06 11 0313 1081
04 A4 00A4 C5 B0 0313 1082
06 11 0313 1083
04 A4 00A4 C5 B0 0313 1084
06 11 0313 1085
04 A4 00A4 C5 B0 0313 1086
06 11 0313 1087
04 A4 00A4 C5 B0 0313 1088
06 11 0313 1089
04 A4 00A4 C5 B0 0313 1090
06 11 0313 1091
04 A4 00A4 C5 B0 0313 1092
06 11 0313 1093
04 A4 00A4 C5 B0 0313 1094
06 11 0313 1095
04 A4 00A4 C5 B0 0313 1096
06 11 0313 1097
04 A4 00A4 C5 B0 0313 1098
06 11 0313 1099
04 A4 00A4 C5 B0 0313 1100
06 11 0313 1101
04 A4 00A4 C5 B0 0313 1102
06 11 0313 1103
04 A4 00A4 C5 B0 0313 1104
06 11 0313 1105
04 A4 00A4 C5 B0 0313 1106
06 11 0313 1107
04 A4 00A4 C5 B0 0313 1108
06 11 0313 1109
04 A4 00A4 C5 B0 0313 1110
06 11 0313 1111
04 A4 00A4 C5 B0 0313 1112
06 11 0313 1113
04 A4 00A4 C5 B0 0313 1114
06 11 0313 1115
04 A4 00A4 C5 B0 0313 1116
06 11 0313 1117
04 A4 00A4 C5 B0 0313 1118
06 11 0313 1119
04 A4 00A4 C5 B0 0313 1120
06 11 0313 1121
04 A4 00A4 C5 B0 0313 1122
06 11 0313 1123
04 A4 00A4 C5 B0 0313 1124
06 11 0313 1125
04 A4 00A4 C5 B0 0313 1126
06 11 0313 1127
04 A4 00A4 C5 B0 0313 1128
06 11 0313 1129
04 A4 00A4 C5 B0 0313 1130
06 11 0313 1131
04 A4 00A4 C5 B0 0313 1132
06 11 0313 1133
04 A4 00A4 C5 B0 0313 1134
06 11 0313 1135
04 A4 00A4 C5 B0 0313 1136
06 11 0313 1137
04 A4 00A4 C5 B0 0313 1138
06 11 0313 1139
04 A4 00A4 C5 B0 0313 1140
06 11 0313 1141
04 A4 00A4 C5 B0 0313 1142
06 11 0313 1143
04 A4 00A4 C5 B0 0313 1144
06 11 0313 1145
04 A4 00A4 C5 B0 0313 1146
06 11 0313 1147
04 A4 00A4 C5 B0 0313 1148
06 11 0313 1149
04 A4 00A4 C5 B0 0313 1150
06 11 0313 1151
04 A4 00A4 C5 B0 0313 1152
06 11 0313 1153
04 A4 00A4 C5 B0 0313 1154
06 11 0313 1155
04 A4 00A4 C5 B0 0313 1156
06 11 0313 1157
04 A4 00A4 C5 B0 0313 1158
06 11 0313 1159
04 A4 00A4 C5 B0 0313 1160
06 11 0313 1161
04 A4 00A4 C5 B0 0313 1162
06 11 0313 1163
04 A4 00A4 C5 B0 0313 1164
06 11 0313 1165
04 A4 00A4 C5 B0 0313 1166
06 11 0313 1167
04 A4 00A4 C5 B0 0313 1168
06 11 0313 1169
04 A4 00A4 C5 B0 0313 1170
06 11 0313 1171
04 A4 00A4 C5 B0 0313 1172
06 11 0313 1173
04 A4 00A4 C5 B0 0313 1174
06 11 0313 1175
04 A4 00A4 C5 B0 0313 1176
06 11 0313 1177
04 A4 00A4 C5 B0 0313 1178
06 11 0313 1179
04 A4 00A4 C5 B0 0313 1180
06 11 0313 1181
04 A4 00A4 C5 B0 0313 1182
06 11 0313 1183
04 A4 00A4 C5 B0 0313 1184
06 11 0313 1185
04 A4 00A4 C5 B0 0313 1186
06 11 0313 1187
04 A4 00A4 C5 B0 0313 1188
06 11 0313 1189
04 A4 00A4 C5 B0 0313 1190
06 11 0313 1191
04 A4 00A4 C5 B0 0313 1192
06 11 0313 1193
04 A4 00A4 C5 B0 0313 1194
06 11 0313 1195
04 A4 00A4 C5 B0 0313 1196
06 11 0313 1197
04 A4 00A4 C5 B0 0313 1198
06 11 0313 1199
04 A4 00A4 C5 B0 0313 1200
06 11 0313 1201
04 A4 00A4 C5 B0 0313 1202
06 11 0313 1203
04 A4 00A4 C5 B0 0313 1204
06 11 0313 1205
04 A4 00A4 C5 B0 0313 1206
06 11 0313 1207
04 A4 00A4 C5 B0 0313 1208
06 11 0313 1209
04 A4 00A4 C5 B0 0313 1210
06 11 0313 1211
04 A4 00A4 C5 B0 0313 1212
06 11 0313 1213
04 A4 00A4 C5 B0 0313 1214
06 11 0313 1215
04 A4 00A4 C5 B0 0313 1216
06 11 0313 1217
04 A4 00A4 C5 B0 0313 1218
06 11 0313 1219
04 A4 00A4 C5 B0 0313 1220
06 11 0313 1221
04 A4 00A4 C5 B0 0313 1222
06 11 0313 1223
04 A4 00A4 C5 B0 0313 1224
06 11 0313 1225
04 A4 00A4 C5 B0 0313 1226
06 11 0313 1227
04 A4 00A4 C5 B0 0313 1228
06 11 0313 1229
04 A4 00A4 C5 B0 0313 1230
06 11 0313 1231
04 A4 00A4 C5 B0 0313 1232
06 11 0313 1233
04 A4 00A4 C5 B0 0313 1234
06 11 0313 1235
04 A4 00A4 C5 B0 0313 1236
06 11 0313 1237
04 A4 00A4 C5 B0 0313 1238
06 11 0313 1239
04 A4 00A4 C5 B0 0313 1240
06 11 0313 1241
04 A4 00A4 C5 B0 0313 1242
06 11 0313 1243
04 A4 00A4 C5 B0 0313 1244
06 11 0313 1245
04 A4 00A4 C5 B0 0313 1246
06 11 0313 1247
04 A4 00A4 C5 B0 0313 1248
06 11 0313 1249
04 A4 00A4 C5 B0 0313 1250
06 11 0313 1251
04 A4 00A4 C5 B0 0313 1252
06 11 0313 1253
04 A4 00A4 C5 B0 0313 1254
06 11 0313 1255
04 A4 00A4 C5 B0 0313 1256
06 11 0313 1257
04 A4 00A4 C5 B0 0313 1258
06 11 0313 1259
04 A4 00A4 C5 B0 0313 1260
06 11 0313 1261
04 A4 00A4 C5 B0 0313 1262
06 11 0313 1263
04 A4 00A4 C5 B0 0313 1264
06 11 0313 1265
04 A4 00A4 C5 B0 0313 1266
06 11 0313 1267
04 A4 00A4 C5 B0 0313 1268
06 11 0313 1269
04 A4 00A4 C5 B0 0313 1270
06 11 0313 1271
04 A4 00A4 C5 B0 0313 1272
06 11 0313 1273
04 A4 00A4 C5 B0 0313 1274
06 11 0313 1275
04 A4 00A4 C5 B0 0313 1276
06 11 0313 1277
04 A4 00A4 C5 B0 0313 1278
06 11 0313 1279
04 A4 00A4 C5 B0 0313 1280
06 11 0313 1281
04 A4 00A4 C5 B0 0313 1282
06 11 0313 1283
04 A4 00A4 C5 B0 0313 1284
06 11 0313 1285
04 A
```


	00000000	'GF	16	0338	729	JSB	G^IOC\$DIAGBUFILL	; Fill diagnostic buffer
	50	8E	D0	033E	730	MOVL	(SP)+,R0	; Get final device status
51	00B0	C5	A5	0341	731	MULW3	#2,UCBSW XA WCR(R5),R1	; Calculate final transfer count
	51	7E	A5	A0	0347	ADDW	UCBSW BCNT(R5),R1	
50	10	10	F0	034B	733	INSV	R1,#16,#16,R0	; Insert into high byte of IOSB
	51	00A8	C5	D0	0350	MOVL	3SW XA CSR(R5),R1	; Return CSR and EIR in IOSB
04	A4	40	8F	88	0355	BISB	*XA_CSR\$M_IE,XA_CSR(R4)	; Enable interrupts
				035A	736	REQCOM		; Finish request in exec


```
0360 738
0360 739 .DSABL LSB
0360 740 :++
0360 741 : WORD MODE -- Process word mode (interrupt per word) transfer
0360 742 :
0360 743 : FUNCTIONAL DESCRIPTION:
0360 744 :
0360 745 : Data is transferred one word at a time with an interrupt for each word.
0360 746 : The request is handled separately for a write (from memory to DR11-W
0360 747 : and a read (from DR11-W to memory).
0360 748 : For a write, data is fetched from memory, loaded into the ODR of the
0360 749 : DR11-W and the system waits for an interrupt. For a read, the system
0360 750 : waits for a DR11-W interrupt and the IDR is transferred into memory.
0360 751 : If the unsolicited interrupt flag is set, the first word is transferred
0360 752 : directly into memory without waiting for an interrupt.
0360 753 :--
0360 754 :
0360 755 .ENABL LSB
0360 756 WORD_MODE:
0360 757
0360 758
0360 759 ; Dispatch to separate loops on READ or WRITE
0360 760
52 0C 91 0360 761 CMPB #IOS_READPBLK,R2 ; Check for read function
6E 13 0363 762 BEQL 30$
0365 763
0365 764 :++
0365 765 : WORD MODE WRITE -- Write (output) in word mode
0365 766 :
0365 767 : FUNCTIONAL DESCRIPTION:
0365 768 :
0365 769 : Transfer the requested number of words from user memory to
0365 770 : the DR11-W ODR one word at a time, wait for interrupt for each
0365 771 : word.
0365 772 :--
0365 773 :
0365 774 10$:
00DA 30 0365 775 BSBW MOVFRUSER ; Get two bytes from user buffer
0368 776 DSBINT ; Lock out interrupts
036E 777 ; Flag interrupt expected
06 A4 51 B0 036E 778 MOVW R1,XA_ODR(R4) ; Move data to DR11-W
04 A4 00A4 C5 B0 0372 779 MOVW UCBSW_XA_CSRTMP(R5),XA_CSR(R4) ; Set DR11-W CSR
07 44 A5 01 E1 0378 780 BCC #XASV_LINK,UCBSL_DEVDEPEND(R5),15$ ; Link mode?
04 A4 00A4 C5 04 AB 037D 781 BICW3 #XASK_FNCT2,UCBSW_XA_CSRTMP(R5),XA_CSR(R4) ; Clear interrupt FNCT bi
0384 782 ; Only if link mode specified
0384 783 15$:
0384 784
0384 785 ; Wait for interrupt, powerfail, or device time-out
0384 786
0384 787 WFIKPCX XA_TIME_OUTW,IRPSL_MEDIA(R3)
038F 788
038F 789 ; Check for errors, decrement transfer count, and loop til complete
038F 790
038F 791 IOFORK ; Fork to lower IPL
00AA C5 5E00 8F B3 0395 792 BITW #XA_EIRSM_NEX!-
039C 793 XA_EIRSM_MULTI!-
039C 794 XA_EIRSM_ACLO!-
```



```
039C 795 XA_EIRSM_PAR!-
039C 796 XA_EIRSM_DLT,UCB$W_XA_EIR(R5) ; Any errors?
039C 797 BEQL 20$ ; No, continue
0087 31 039E 798 BRW 40$ ; Yes, abort transfer.
00B4 C5 B7 03A1 799 20$: DECW UCB$X_XA_DPR(R5) ; All words trnasferred?
BE 12 03A5 800 BNEQ 10$ ; No, loop until finished.
03A7 801
03A7 802 ; Transfer is done, clear interrupt expected flag and FORK
03A7 803 ; All words read or written in WORD MODE. Finish I/O.
03A7 804
03A7 805 RETURN_STATUS:
03A7 806
00000000'GF 16 03A7 807 JSB G^IOC$DIAGBUFILL ; Fill diagnostic buffer if present
01DC 30 03AD 808 BSBW DEL_ATTNAST ; Deliver outstanding ATTN AST's
51 50 01 3C 03B0 809 MOVZWL #SS$ NORMAL,R0 ; Complete success status
51 00B4 C5 02 A5 03B3 810 22$: MULW3 #2,UCB$X_XA_DPR(R5),R1 ; Calculate actual bytes xfered
51 7E A5 51 A3 03B9 811 SUBW3 R1,UCB$W_BCNT(R5),R1 ; From requested number of bytes
50 10 10 51 F0 03BE 812 INSV R1,#16,#T6,R0 ; And place in high word of R0
51 00A8 C5 D0 03C3 813 MOVL UCB$W_XA_CSR(R5),R1 ; Return CSR and EIR status
04 A4 40 8F 88 03C8 814 BISB #XA_CSR$M_IE,XA_CSR(R4) ; Enable device interrupts
03CD 815 REQCOM ; Finish request in exec
03D3 816
03D3 817 ;++
03D3 818 ; WORD MODE READ -- Read (input) in word mode
03D3 819
03D3 820 ; FUNCTIONAL DESCRIPTION:
03D3 821
03D3 822 ; Transfer the requested number of wrods from the DR11-W IDR into
03D3 823 ; user memory one word at a time, wait for interrupt for each word.
03D3 824 ; If the unexpected (unsolicited) interrupt bit is set, transfer the
03D3 825 ; first (last received) word to memory without waiting for an
03D3 826 ; interrupt.
03D3 827 ;--
03D3 828
03D3 829 30$:
03D3 830 DSBINT UCB$B_DIPL(R5) ; Lock out interrupts
03DA 831
03DA 832 ; If an unexpected (unsolicited) interrupt has occured, assume it
03DA 833 ; is for this READ request and return value to user buffer without
03DA 834 ; waiting for an interrupt.
03DA 835
05 01 E5 03DA 836 BBCC #UCB$V_UNEXPT,- ; Branch if no unexpected interrupt
05 68 A5 03DC 837 UCB$W_DEVSTS(R5),32$ ; Enable interrupts
03DF 838 ENBINT ; continue
14 11 03E2 839 BRB 37$
03E4 840
03E4 841 32$:
03E4 842 SETIPL #IPL$_POWER
03E7 843 35$:
03E7 844
03E7 845 ; Wait for interrupt, powerfail, or device time-out
03E7 846
03E7 847 WFIKPBH XA_TIME_OUTW,IRP$X_MEDIA(R3)
03F2 848
03F2 849 ; Check for errors, decrement transfer count and loop until done
03F2 850
03F2 851 IOFORK ; Fork to lower IPL
```



```
00AA C5 5E00 8F B3 03F8 852 37$:
03F8 853 BITW #XA_EIRSM_NEX!-
03FF 854 XA_EIRSM_MULTI!-
03FF 855 XA_EIRSM_ACLO!-
03FF 856 XA_EIRSM_PAR!-
03FF 857 XA_EIRSM_DLT,UCBSW_XA_EIR(R5) ; Any errors?
27 12 03FF 858 BNEQ 40$ ; Yes, abort transfer.
004F 30 0401 859 BSBW MOVTOUSER ; Store two bytes into user buffer
0404 860
0404 861 ; Send interrupt back to sender. Acknowledge we got last word.
0404 862
0404 863 DSBINT
040A 864 MOVW UCBSW_XA_CSRTMP(R5),XA_CSR(R4)
07 44 A5 01 E1 0410 865 BBC #XASV_LINK,UCBSL_DEVDEPEND(R5),38$ ; Link mode?
04 A4 00A4 C5 04 AB 0415 866 BICW3 #XASK_FNCT2,UCBSW_XA_CSRTMP(R5),XA_CSR(R4) ; Yes, clear FNCT 2
041C 867 38$:
00B4 C5 B7 041C 868 DECW UCBSL_XA_DPR(R5) ; Decrement transfer count
C5 12 0420 869 BNEQ 35$ ; Loop until all words transferred
0422 870 ENBINT
FF7F 31 0425 871 BRW RETURN_STATUS ; Finish request in common code
0428 872
0428 873 ; Error detected in word mode transfer
0428 874
0428 875 40$:
0155 30 0428 876 BSBW DEL_ATTNAST ; Deliver ATTN AST's
01D0 30 042B 877 BSBW XA_DEV_RESET ; Error, reset DR11-W
00000000'GF 16 042E 878 JSB G^IOCS$DIAGBUFILL ; Fill diagnostic buffer if presetn
00000000'GF 16 0434 879 JSB G^ERL$DEVICERR ; Log device error
50 00B2 C5 3C 043A 880 MOVZWL UCBSW_XA_ERROR(R5),R0 ; Set controller/drive status in R0
FF71 31 043F 881 BRW 22$
0442 882
0442 883 .DSABL LSB
0442 884
0442 885 : MOVFRUSER - Routine to fetch two bytes from user buffer.
0442 886 :
0442 887 : INPUTS:
0442 888 :
0442 889 : R5 = UCB address
0442 890 :
0442 891 : OUTPUTS:
0442 892 :
0442 893 : R1 = Two bytes of data from users buffer
0442 894 : Buffer descriptor in UCB is updated.
0442 895 :
0442 896 : .ENABL LSB
0442 897 MOVFRUSER:
51 7E DE 0442 898 MOVAL -(SP),R1 ; Address of temporary stack loc
52 02 9A 0445 899 MOVZBL #2,R2 ; Fetch two bytes
00000000'GF 16 0448 900 JSB G^IOCS$MOVFRUSER ; Call exec routine to do the deed
51 8E D0 044E 901 MOVL (SP)+,R1 ; Retrieve the bytes
OE 11 0451 902 BRB 20$ ; Update UCB buffer pointers
0453 903 :
0453 904 : MOVTOUSER - Routine to store two bytes into users buffer.
0453 905 :
0453 906 : INPUTS:
0453 907 :
0453 908 : R5 = UCB address
```



```
0453 909 :      UCB$W_XA_IDR(R5) = Location where two bytes are saved
0453 910 :
0453 911 :      OUTPUTS:
0453 912 :
0453 913 :      Two bytes are stored in user buffer and buffer descriptor in
0453 914 :      UCB is updated.
0453 915 :
0453 916 MOVTOUSER:
51 00AC C5 9E 0453 917 MOVAB UCB$W_XA_IDR(R5),R1 ; Address of internal buffer
52 02 9A 0458 918 MOVZBL #2,R2
00000000'GF 16 045B 919 JSB G^IOC$MOVTOUSER ; Call exec
7C A5 02 A0 0461 920 20$: ; Update buffer pointers in UCB
7C A5 FE00 8F AA 0461 921 ADDW #2,UCB$W_BOFF(R5) ; Add two to buffer descriptor
78 A5 04 12 0465 922 BICW #^C<^X01FF>,UCB$W_BOFF(R5) ; Modulo the page size
05 046B 923 BNEQ 30$ ; If NEQ, no page boundary crossed
046D 924 ADDL #4,UCB$L_SVAPTE(R5) ; Point to next page
0471 925 30$:
0471 926 RSB
0472 927 ;
0472 928 .DSABL LSB
```



```
0472 931 .SBTTL DR11-W DEVICE TIME-OUT
0472 932 :++
0472 933 : DR11-W device TIME-OUT
0472 934 : If a DMA transfer was in progress, release UBA resources.
0472 935 : For DMA or WORD mode, deliver ATTN AST's, log a device timeout error,
0472 936 : and do a hard reset on the controller.
0472 937 :
0472 938 : Clear DR11-W CSR
0472 939 : Return error status
0472 940 :
0472 941 : Power failure will appear as a device time-out
0472 942 :--
0472 943 .ENABL LSB
0472 944 XA_TIME_OUT: ; Time-out for DMA transfer
0472 945
0472 946 SETIPL UCB$B_FIPL(R5) ; Lower to FORK IPL
0476 947 PURDPR ; Purge buffered data path in UBA
047C 948 RELMPR ; Release UBA map registers
0482 949 RELDPR ; Release UBA data path
0488 950 BRB 10$ ; continue
048A 951
048A 952 XA_TIME_OUTW: ; Time-out for WORD mode transfer
048A 953
048A 954 SETIPL UCB$B_FIPL(R5) ; Lower to FORK IPL
54 24 A5 D0 048E 955 10$: MOVL UCB$L_CRB(R5),R4 ; Fetch address of CSR
54 2C B4 D0 0492 956 MOVL @CRB$C_INTD+VEC$L_IDB(R4),R4
005C 30 0496 957 BSBW XA_REGISTER ; Read DR11-W registers
00000000'GF 16 0499 958 JSB G^IOC$DIAGBUFILL ; Fill diagnostic buffer
00000000'GF 16 049F 959 JSB G^ERL$DEVICTMO ; Log device time out
00D8 30 04A5 960 BSBW DEL_ATTNAST ; And deliver the AST's
0153 30 04A8 961 BSBW XA_DEV_RESET ; Reset controller
50 022C 8F 3C 04AB 962 MOVZWL #$$$_TIMEOUT,R0 ; Assume error status
03 E1 04B0 963 BBC #UCB$V_CANCEL,-
05 64 A5 3C 04B2 964 UCB$W_STS(R5),20$ ; Branch if not cancel
50 0830 8F 3C 04B5 965 MOVZWL #$$$_CANCEL,R0 ; Set status
51 D4 04BA 966 20$: CLRL R1
68 A5 B4 04BC 967 CLRW UCB$W_DEVSTS(R5) ; Clear ATTN AST flags
006B 8F AA 04BF 968 BICW #<UCB$M_TIM!UCB$M_INT!UCB$M_TIMEOUT!UCB$M_CANCEL!UCB$M_POWER>,-
64 A5 04C3 969 UCB$W_STS(R5) ; Clear unit status flags
04C5 970 REQCOM ; Complete I/O in exec
04CB 971 .DSABL LSB
```



```
04CB 974      .SBTTL  XA_INTERRUPT, Interrupt service routine for DR11-W
04CB 975      :++
04CB 976      : XA_INTERRUPT, Handles interrupts generated by DR11-W
04CB 977      :
04CB 978      : Functional description:
04CB 979      :
04CB 980      : This routine is entered whenever an interrupt is generated
04CB 981      : by the DR11-W. It checks that an interrupt was expected.
04CB 982      : If not, it sets the unexpected (unsolicited) interrupt flag.
04CB 983      : All device registers are read and stored into the UCB.
04CB 984      : If an interrupt was expected, it calls the driver back at its Wait
04CB 985      : For Interrupt point.
04CB 986      : Deliver ATTN AST's if unexpected interrupt.
04CB 987      :
04CB 988      : Inputs:
04CB 989      :
04CB 990      : 00(SP) = Pointer to address of the device IDB
04CB 991      : 04(SP) = saved R0
04CB 992      : 08(SP) = saved R1
04CB 993      : 12(SP) = saved R2
04CB 994      : 16(SP) = saved R3
04CB 995      : 20(SP) = saved R4
04CB 996      : 24(SP) = saved R5
04CB 997      : 28(SP) = saved PSL
04CB 998      : 32(SP) = saved PC
04CB 999      :
04CB 1000     : Outputs:
04CB 1001     :
04CB 1002     : The driver is called at its Wait For Interrupt point if an
04CB 1003     : interrupt was expected.
04CB 1004     : The current value of the DR11-W CSR's are stored in the UCB.
04CB 1005     :
04CB 1006     :--
04CB 1007     XA_INTERRUPT:                                ; Interrupt service for DR11-W
54 9E D0 04CB 1008     MOVL    @ (SP)+, R4                    ; Address of IDB and pop SP
54 64 7D 04CE 1009     MOVQ    (R4), R4                      ; CSR and UCB address from IDB
04D1 1010
04D1 1011     ; Read the DR11-W device registers (WCR, BAR, CSR, EIR, IDR) and store
04D1 1012     ; into UCB.
04D1 1013
0021 30 04D1 1014     BSBW     XA_REGISTER                    ; Read device registers
04D4 1015
04D4 1016     ; Check to see if device transfer request active or not
04D4 1017     ; If so, call driver back at Wait for Interrupt point and
04D4 1018     ; Clear unexpected interrupt flag.
04D4 1019
0D 64 A5 01 E5 04D4 1020 20$: BBCC    #UCB$V_INT,UCB$W_STS(R5),25$
04D9 1021                                           ; If clear, no interrupt expected
04D9 1022
04D9 1023     ; Interrupt expected, clear unexpected interrupt flag and call driver
04D9 1024     ; back.
04D9 1025
68 A5 02 AA 04D9 1026     BICW    #UCB$M_UNEXPT,UCB$W_DEVSTS(R5)
04DD 1027                                           ; Clear unexpected interrupt flag
53 10 A5 D0 04DD 1028     MOVL    UCB$L_FR3(R5), R3          ; Restore drivers R3
OC B5 16 04E1 1029     JSB     @UCB$L_FPC(R5)                ; Call driver back
OC 11 04E4 1030     BRB     30$
```



```

04E6 1031
04E6 1032 ; Deliver ATTN AST's if no interrupt expected and set unexpected
04E6 1033 ; interrupt flag.
04E6 1034
04E6 1035 25$:
68 A5 02 A8 04E6 1036 BISW #UCB$M_UNEXPT,UCB$W_DEVSTS(R5) ; Set unexpected interrupt flag
0093 30 04EA 1037 BSBW DEL_ATTNA$T ; Deliver ATTN AST's
04 A4 40 8F 88 04ED 1038 BISB #XA_CSR$M_IE,XA_CSR(R4) ; Enable device interrupts
04F2 1039
04F2 1040 ; Restore registers and return from interrupt
04F2 1041
04F2 1042 30$:
3F BA 04F2 1043 POPR #^M<R0,R1,R2,R3,R4,R5> ; Restore registers
02 04F4 1044 REI ; Return from interrupt
```



```
04F5 1047 .SBTTL XA_REGISTER - Handle DR11-W CSR transfers
04F5 1048 :++
04F5 1049 : XA_REGISTER - Routine to handle DR11-W register transfers
04F5 1050 :
04F5 1051 : INPUTS:
04F5 1052 :
04F5 1053 :     R4 - DR11-W CSR address
04F5 1054 :     R5 - UCB address of unit
04F5 1055 :
04F5 1056 : OUTPUTS:
04F5 1057 :
04F5 1058 :     CSR, EIR, WCR, BAR, IDR, and status are read and stored into UCB.
04F5 1059 :     The DR11-W is placed in its initial state with interrupts enabled.
04F5 1060 :     R0 - .true. if no hard error
04F5 1061 :         .false. if hard error (cannot clear ATTN)
04F5 1062 :
04F5 1063 :     If the CSR ERROR bit is set and the associated condition can be cleared, then
04F5 1064 :     the error is transient and recoverable. The status returned is SSS$DRVERR.
04F5 1065 :     If the CSR ERROR bit is set and cannot be cleared by clearing the CSR, then
04F5 1066 :     this is a hard error and cannot be recovered. The returned status is
04F5 1067 :     SSS$CTRLERR.
04F5 1068 :
04F5 1069 :     R0,R1 - destroyed, all other registers preserved.
04F5 1070 :--
04F5 1071 :
04F5 1072 XA_REGISTER:
04F5 1073
04F5 1074 MOV2WL #SS$ NORMAL,R0 ; Assume success
04F5 1075 MOVZWL XA_CSR(R4),R1 ; Read CSR
04F5 1076 MOVW R1,UCB$W_XA_CSR(R5) ; Save CSR in UCB
04F5 1077 BBC #XA_CSR$V_ERROR,R1,55$ ; Branch if no error
04F5 1078 MOVZWL #SS$DRVERR,R0 ; Assume "drive" error
04F5 1079 55$: BICW #^C<XA_CSR$M_FNCT>,R1 ; Clear all uninteresting bits for later
04F5 1080 BISB #<XA_CSR$M_ERROR/256>,XA_CSR+1(R4) ; Set EIR flag
04F5 1081 MOVW XA_EIR(R4),UCB$W_XA_EIR(R5) ; Save EIR in UCB
04F5 1082 MOVW R1,XA_CSR(R4) ; Clear EIR flag and errors
04F5 1083 MOVW XA_CSR(R4),R1 ; Read CSR back
04F5 1084 BBC #XA_CSR$V_ATTN,R1,60$ ; If attention still set, hard error
04F5 1085 MOVZWL #SS$CTRLERR,R0 ; Flag hard controller error
04F5 1086 60$: MOVW XA_IDR(R4),UCB$W_XA_IDR(R5) ; Save IDR in UCB
04F5 1087 MOVW XA_BAR(R4),UCB$W_XA_BAR(R5)
04F5 1088 MOVW XA_WCR(R4),UCB$W_XA_WCR(R5)
04F5 1089 MOVW R0,UCB$W_XA_ERROR(R5) ; Save status in UCB
04F5 1090 RSB
```

50	01	3C	04F5	1074
51	04	A4	3C	04F8
00A8	C5	51	B0	04FC
05	51	0F	E1	0501
50	008C	8F	3C	0505
51	FFF1	8F	AA	050A
05	A4	80	88	050F
00AA	C5	04	B0	0514
04	A4	51	B0	051A
51	04	A4	B0	051E
05	51	0D	E1	0522
50	0054	8F	3C	0526
00AC	C5	06	B0	052B
00AE	C5	02	B0	0531
00B0	C5	64	B0	0537
00B2	C5	50	B0	053C
			05	0541


```
0542 1092 .SBTTL XA_CANCEL, Cancel I/O routine
0542 1093 :++
0542 1094 : XA_CANCEL, Cancels an I/O operation in progress
0542 1095 :
0542 1096 : Functional description:
0542 1097 :
0542 1098 :     Flushes Attention AST queue for the user.
0542 1099 :     If transfer in progress, do a device reset to DR11-W and finish the
0542 1100 :     request.
0542 1101 :     Clear interrupt expected flag.
0542 1102 :
0542 1103 : Inputs:
0542 1104 :
0542 1105 :     R2 = negated value of channel index
0542 1106 :     R3 = address of current IRP
0542 1107 :     R4 = address of the PCB requesting the cancel
0542 1108 :     R5 = address of the device's UCB
0542 1109 :
0542 1110 : Outputs:
0542 1111 :
0542 1112 :--
0542 1113
0542 1114 XA_CANCEL:                                ; Cancel I/O
0542 1115
0542 1116 BBCC #UCB$V ATTNAST, -
16 68 A5 E5 0544 1117 UCB$W_DEVSTS(R5),20$ ; ATTN AST enabled?
0547 1118
0547 1119 ; Finish all ATTN AST's for this process.
0547 1120
0547 1121 PUSHR #^M<R2,R6,R7>
054B 1122 MOVL R2,R6 ; Set up channel number
57 00A0 C5 9E 054E 1123 MOVAB UCB$L_XA_ATTN(R5),R7 ; Address of listhead
00000000'GF 16 0553 1124 JSB G^COM$FLUSHATTNS ; Flush ATTN AST's for process
00C4 8F BA 0559 1125 POPR #^M<R2,R6,R7>
055D 1126
055D 1127 ; Check to see if a data transfer request is in progress
055D 1128 ; for this process on this channel
055D 1129
055D 1130 20$:
055D 1131 DSBINT UCB$B_DIPL(R5) ; Lock out device interrupts
00000000'GF 16 0564 1132 JSB G^IOC$CANCELIO ; Check if transfer going
03 056A 1133 BBC #UCB$V_CANCEL, -
0D 64 A5 E1 056C 1134 UCB$W_STS(R5),30$ ; Branch if not for this guy
056F 1135
056F 1136 ; Force timeout
056F 1137
056F 1138 CLRL UCB$L_DUETIM(R5) ; clear timer
64 A5 01 D4 0572 1139 BISW #UCB$M_TIM,UCB$W_STS(R5) ; set timed
0040 8F A8 0576 1140 BICW #UCB$M_TIMEOUT, -
64 A5 AA 057A 1141 UCB$W_STS(R5) ; Clear timed out
057C 1142 30$:
057C 1143 ENBINT ; Lower to FORK IPL
05 057F 1144 RSB ; Return
```



```
0580 1147 .SBTTL DEL_ATTNAST, Deliver ATTN AST's
0580 1148 :++
0580 1149 : DEL_ATTNAST, Deliver all outstanding ATTN AST's
0580 1150 :
0580 1151 : Functional description:
0580 1152 :
0580 1153 : This routine is used by the DR11-W driver to deliver all of the
0580 1154 : outstanding attention AST's. It is copied from COM$DELATTNAST in
0580 1155 : the exec. In addition, it places the saved value of the DR11-W CSR
0580 1156 : and Input Data Buffer Register in the AST paramater.
0580 1157 :
0580 1158 : Inputs:
0580 1159 :
0580 1160 : R5 = UCB of DR11-W unit
0580 1161 :
0580 1162 : Outputs:
0580 1163 :
0580 1164 : R0,R1,R2 Destroyed
0580 1165 : R3,R4,R5 Preserved
0580 1166 :--
0580 1167 DEL_ATTNAST:
0580 1168 DSBIN1 UCB$B_DIPL(R5) ; Device IPL
0587 1169 BBCC #UCB$V_ATTNAST,UCB$W_DEVSTS(R5),30$ ;
058C 1170 ; Any ATTN AST's expected?
058C 1171 PUSHR #^M<R3,R4,R5> ; Save R3,R4,R5
058E 1172 10$: MOVL 8(SP),R1 ; Get address of UCB
0592 1173 MOVAB UCB$A_XA_ATTNAST(R1),R2 ; Address of ATTN AST listhead
0597 1174 MOVL (R2),R5 ; Address of next entry on list
059A 1175 BEQL 20$ ; No next entry, end of loop
059C 1176 BICW #UCB$M_UNEXPT,UCB$W_DEVSTS(R1) ; Clear unexpected interrupt flag
05A0 1177 MOVL (R5),(R2) ; Close list
05A3 1178 MOVW UCB$W_XA_IDR(R1),ACB$A_KAST+6(R5) ; Store IDR in AST paramater
05A9 1179 MOVW UCB$W_XA_CSR(R1),ACB$A_KAST+4(R5) ; Store CSR in AST paramater
05AF 1181 PUSHAB B^10$ ; Set return address for FORK
05AF 1182 FORK ; FORK for this AST
05B2 1183
05B8 1184
05B8 1185 ; AST fork procedure
05B8 1186
05B8 1187 MOVQ AKB$A_KAST(R5),AKB$A_AST(R5) ; Re-arrange entries
05BD 1188
05BD 1189 MOVAB AKB$A_KAST+8(R5),AKB$B_RMOD(R5)
05C2 1190 MOVL AKB$A_KAST+12(R5),AKB$C_PID(R5)
05C7 1191 CLRL AKB$A_KAST(R5)
05CA 1192 MOVZBL #PRI$-IOCOM,R2 ; Set up priority increment
05CD 1193 JMP G^SCH$QAST ; Queue the AST
05D3 1194
05D3 1195 20$: POPR #^M<R3,R4,R5> ; Restore registers
05D5 1196 30$: ENBINT ; Enable interrupts
05D8 1197 RSB ; Return
```



```
05D9 1200      .SBTTL  XA_REGDUMP - DR11-W register dump routine
05D9 1201      :++
05D9 1202      : XA_REGDUMP - DR11-W Register dump routine.
05D9 1203      :
05D9 1204      : This routine is called to save the controller registers in a specified
05D9 1205      : buffer. It is called from the device error logging routine and from the
05D9 1206      : diagnostic buffer fill routine.
05D9 1207      :
05D9 1208      : Inputs:
05D9 1209      :
05D9 1210      :     R0 - Address of register save buffer
05D9 1211      :     R4 - Address of Control and Status Register
05D9 1212      :     R5 - Address of UCB
05D9 1213      :
05D9 1214      : Outputs:
05D9 1215      :
05D9 1216      :     The controller registers are saved in the specified buffer.
05D9 1217      :
05D9 1218      :     CSRTMP - The last command written to the DR11-W CSR by
05D9 1219      :             by the driver.
05D9 1220      :     BARTMP - The last value written into the DR11-W BAR by
05D9 1221      :             the driver during a block mode transfer.
05D9 1222      :     CSR - The CSR image at the last interrupt
05D9 1223      :     EIR - The EIR image at the last interrupt
05D9 1224      :     IDR - The IDR image at the last interrupt
05D9 1225      :     BAR - The BAR image at the last interrupt
05D9 1226      :     WCR - Word count register
05D9 1227      :     ERROR - The system status at request completion
05D9 1228      :     PDRN - UBA Datapath Register number
05D9 1229      :     DPR - The contents of the UBA Data Path register
05D9 1230      :     FMPR - The contents of the last UBA Map register
05D9 1231      :     PMRP - The contents of the previous UBA Map register
05D9 1232      :     DPRF - Flag for purge datapath error
05D9 1233      :             0 = no purger datapath error
05D9 1234      :             1 = parity error when datapath was purged
05D9 1235      :
05D9 1236      : Note that the values stored are from the last completed transfer
05D9 1237      : operation. If a zero transfer count is specified, then the
05D9 1238      : values are from the last operation with a non-zero transfer count.
05D9 1239      :--
```

```
05D9 1240      :
05D9 1241      : XA_REGDUMP:
05D9 1242      :
51      80      0B      9A      05D9 1243      MOVZBL #11,(R0)+ ; Eleven registers are stored.
      00A4 C5      9E      05DC 1244      MOVAB  UCB$W_XA_CSRTMP(R5),R1 ; Get address of saved register images
      52      08      9A      05E1 1245      MOVZBL #8,R2 ; Return 8 registers here
      80      81      3C      05E4 1246 10$: MOVZWL (R1)+,(R0)+
      FA      52      F5      05E7 1247      SOBGTR R2,10$ ; Move them all
80      00C0 C5      9A      05EA 1248      MOVZBL UCB$W_XA_DPRN(R5),(R0)+ ; Save Datapath Register number
      52      03      9A      05EF 1249      MOVZBL #3,R2 ; And 3 more here
      80      81      D0      05F2 1250 20$: MOVL (R1)+,(R0)+ ; Move UBA register contents
      FA      52      F5      05F5 1251      SOBGTR R2,20$
80      00C1 C5      9A      05F8 1252      MOVZBL UCB$W_XA_DPRN+1(R5),(R0)+ ; Save Datapath Parity Error Flag
      05      05FD 1253      RSB
```



```
05FE 1256 .SBTTL XA_DEV_RESET - Device reset DR11-W
05FE 1257 :++
05FE 1258 : XA_DEV_RESET - DR11-W Device reset routine
05FE 1259 :
05FE 1260 : This routine raises IPL to device IPL, performs a device reset to
05FE 1261 : the required controller, and re-enables device interrupts.
05FE 1262 :
05FE 1263 : Inputs:
05FE 1264 :
05FE 1265 :     R4 - Address of Control and Status Register
05FE 1266 :     R5 - Address of UCB
05FE 1267 :
05FE 1268 : Outputs:
05FE 1269 :
05FE 1270 :     Controller is reset, controller interrupts are enabled
05FE 1271 :
05FE 1272 :--
05FE 1273
05FE 1274 XA_DEV_RESET:
05FE 1275
05FE 1276     PUSH    #M<R0,R1,R2>          ; Save some registers
0600 1277     DSBINT                    ; Raise IPL to lock all interrupts
05 A4 10 90 0606 1278     MOV     #<XA_CSR$M_MAINT/256>,XA_CSR+1(R4)
05 A4 94 060A 1279     CLRB     XA_CSR+1(R4)
060D 1280
060D 1281 : *** Must delay here depending on reset interval
060D 1282
060D 1283     TIMEDWAIT TIME=#XA_RESET_DELAY ; No. of 10 micro-sec intervals to wait
062B 1284
04 A4 40 8F 90 062B 1285     MOV     #XA_CSR$M_IE,XA_CSR(R4) ; Re-enable device interrupts
0630 1286     ENBINT                    ; Restore IPL
07 BA 0633 1287     POPR     #M<R0,R1,R2>          ; Restore registers
0635 1288
05 0635 1289     RSB
0636 1290
0636 1291 XA_END:
0636 1292     .END ; End of driver label
```


XADriver
Symbol table

- VAX/VMS DR11-W DRIVER

D 2

16-SEP-1984 00:14:45 VAX/VMS Macro V04-00
6-SEP-1984 16:32:52 [DRIVER.SRC]XADriver.MAR;2

Page 31
(16)

\$\$\$	= 00000020	R	02	IOSV_TIMED	= 00000007		
\$\$OP	= 00000002			IOSV_WORD	= 00000006		
ACBSB_RMOD	= 0000000B			IOS_READBLK	= 00000021		
ACBSL_AST	= 00000010			IOS_READPBLK	= 0000000C		
ACBSL_KAST	= 00000018			IOS_READVBLK	= 00000031		
ACBSL_PID	= 0000000C			IOS_SENSECHAR	= 0000001B		
ATS_UBA	= 00000001			IOS_SENSEMODE	= 00000027		
BLOCK_MODE	00000226	R	03	IOS_SETCCHAR	= 0000001A		
COMSF_CUSHATTNS	*****	X	03	IOS_SETMODE	= 00000023		
COMSSETATTNAST	*****	X	03	IOS_VIRTUAL	= 0000003F		
CRBSL_INTD	= 00000024			IOS_WRITEBLK	= 00000020		
DCS_REALTIME	= 00000060			IOS_WRITEPBLK	= 0000000B		
DDBSL_DDT	= 0000000C			IOS_WRITEVBLK	= 00000030		
DEL_ATTNAST	00000580	R	03	IOCS_CANCELIO	*****	X	03
DEVSM_AVL	= 00040000			IOCS_DIAGBUFILL	*****	X	03
DEVSM_ELQ	= 00400000			IOCS_LOADUBAMAP	*****	X	03
DEVSM_IDV	= 04000000			IOCS_MNTVER	*****	X	03
DEVSM_ODV	= 08000000			IOCS_MOVFRUSER	*****	X	03
DEVSM_RTM	= 20000000			IOCS_MOVTOUSER	*****	X	03
DPTSC_LENGTH	= 00000038			IOCS_PURGDATA	*****	X	03
DPTSC_VERSION	= 00000004			IOCS_RELDATA	*****	X	03
DPTSINITAB	00000038	R	02	IOCS_RELMAPREG	*****	X	03
DPTSM_SVP	= 00000002			IOCS_REQCOM	*****	X	03
DPTSREINITAB	00000054	R	02	IOCS_REQDATA	*****	X	03
DPTSTAB	00000000	R	02	IOCS_REQMAPREG	*****	X	03
DTS_DR11W	= 00000004			IOCS_RETURN	*****	X	03
DYNSC_CRB	= 00000005			IOCS_WFIKPC	*****	X	03
DYNSC_DDB	= 00000006			IPLS_POWER	= 0000001F		
DYNSC_DPT	= 0000001E			IRPSC_MEDIA	= 00000038		
DYNSC_UCB	= 00000010			IRPSL_SEGVBN	= 00000048		
EMBSL_DV_REGSAV	= 0000004E			IRPSW_FUNC	= 00000020		
ERLSDEVICERR	*****	X	03	MASKH	= 00000080		
ERLSDEVICTMO	*****	X	03	MASKL	= 08000000		
EXESABORTIO	*****	X	03	MOVFRUSER	00000442	R	03
EXESFINISHIOC	*****	X	03	MOVTOUSER	00000453	R	03
EXESFORK	*****	X	03	P1	= 00000000		
EXESGL_TENUSEC	*****	X	03	P2	= 00000004		
EXESGL_UBDELAY	*****	X	03	P3	= 00000008		
EXESIOFORK	*****	X	03	P4	= 0000000C		
EXESMODIFY	*****	X	03	P5	= 00000010		
EXESREAD	*****	X	03	P6	= 00000014		
EXESSENSEMODE	*****	X	03	PR\$ IPL	= 00000012		
EXESSETCHAR	*****	X	03	PRIS IOCOM	= 00000001		
EXESSETMODE	*****	X	03	RETURN STATUS	000003A7	R	03
EXESWRITE	*****	X	03	SCHSQAST	*****	X	03
FUNCTAB_LEN	= 0000004C			SIZ...	= 00000001		
IDBSL_CSR	= 00000000			SS\$_BADPARAM	= 00000014		
IDBSL_OWNER	= 00000004			SS\$_CANCEL	= 00000830		
IDBSL_UCBLST	= 00000018			SS\$_CTRLERR	= 00000054		
IOSS_FCODE	= 00000006			SS\$_DRVERR	= 0000008C		
IOSV_ATTNAST	= 00000008			SS\$_NOPRIV	= 00000024		
IOSV_CYCLE	= 0000000C			SS\$_NORMAL	= 00000001		
IOSV_DATAPATH	= 0000000A			SS\$_OPINCOMPL	= 000002D4		
IOSV_DIAGNOSTIC	= 00000008			SS\$_PARITY	= 000001F4		
IOSV_FCODE	= 00000000			SS\$_TIMEOUT	= 0000022C		
IOSV_RESET	= 0000000B			UCBSB_DEVCLASS	= 00000040		
IOSV_SETFNCT	= 00000009			UCBSB_DEVTYPE	= 00000041		

XADriver
Symbol table

- VAX/VMS DR11-W DRIVER

E 2

16-SEP-1984 00:14:45 VAX/VMS Macro V04-00
6-SEP-1984 16:32:52 [DRIVER.SRC]XADriver.MAR;2

Page 32
(16)

UCBSB_DIPL	= 0000005E		
UCBSB_FIPL	= 0000000B		
UCBSK_SIZE	= 000000C2		
UCBSL_CRB	= 00000024		
UCBSL_DEVCHAR	= 00000038		
UCBSL_DEVDEPEND	= 00000044		
UCBSL_DPC	= 0000009C		
UCBSL_DUETIM	= 0000006C		
UCBSL_FPC	= 0000000C		
UCBSL_FR3	= 00000010		
UCBSL_SVAPTE	= 00000078		
UCBSL_XA_ATTN	000000A0		
UCBSL_XA_DPR	000000B4		
UCBSL_XA_FMPR	000000B8		
UCBSL_XA_PMPR	000000BC		
UCBSM_ATTNAST	= 00000001		
UCBSM_CANCEL	= 00000008		
UCBSM_INT	= 00000002		
UCBSM_ONLINE	= 00000010		
UCBSM_POWER	= 00000020		
UCBSM_TIM	= 00000001		
UCBSM_TIMEOUT	= 00000040		
UCBSM_UNEXPT	= 00000002		
UCBSV_ATTNAST	= 00000000		
UCBSV_CANCEL	= 00000003		
UCBSV_INT	= 00000001		
UCBSV_POWER	= 00000005		
UCBSV_UNEXPT	= 00000001		
UCBSW_BCNT	= 0000007E		
UCBSW_BOFF	= 0000007C		
UCBSW_DEVBUSIZ	= 00000042		
UCBSW_DEVSTS	= 00000068		
UCBSW_FUNC	= 0000009A		
UCBSW_STS	= 00000064		
UCBSW_XA_BAR	000000AE		
UCBSW_XA_BARTMP	000000A6		
UCBSW_XA_CSR	000000A8		
UCBSW_XA_CSRTMP	000000A4		
UCBSW_XA_DPRN	000000C0		
UCBSW_XA_EIR	000000AA		
UCBSW_XA_ERROR	000000B2		
UCBSW_XA_IDR	000000AC		
UCBSW_XA_WCR	000000B0		
VECSB_DATAPATH	= 00000013		
VECSL_IDB	= 00000008		
VECSL_INITIAL	= 0000000C		
VECSM_PATHLOCK	= 00000080		
VECSS_DATAPATH	= 00000005		
VECSS_MAPREG	= 0000000F		
VECSV_DATAPATH	= 00000000		
VECSV_MAPREG	= 00000000		
VECSW_MAPREG	= 00000010		
WORD_MODE	00000360	R	03
XASDDT	00000000	RG	03
XASK_FNCT2	= 00000004		
XASV_DATAPATH	= 00000000		
XASV_LINK	= 00000001		

XA_BAR	00000002		
XA_CANCEL	00000542	R	03
XA_CONTROL_INIT	00000084	R	03
XA_CSR	00000004		
XA_CSRSM_ATTN	= 00002000		
XA_CSRSM_CYCLE	= 00000100		
XA_CSRSM_ERROR	= 00008000		
XA_CSRSM_FNCT	= 0000000E		
XA_CSRSM_GO	= 00000001		
XA_CSRSM_IE	= 00000040		
XA_CSRSM_MAINT	= 00001000		
XA_CSRSV_ATTN	= 0000000D		
XA_CSRSV_ERROR	= 0000000F		
XA_CSRSV_FNCT	= 00000001		
XA_CSRSV_XBA	= 00000004		
XA_DEF_BUFSIZ	= 0000FFFF		
XA_DEF_TIMEOUT	= 0000000A		
XA_DEV_RESET	000005FE	R	03
XA_EIR	00000004		
XA_EIRSM_ACLO	= 00000800		
XA_EIRSM_DLT	= 00000200		
XA_EIRSM_MULTI	= 00001000		
XA_EIRSM_NEX	= 00004000		
XA_EIRSM_PAR	= 00000400		
XA_END	00000636	R	03
XA_FUNCTABLE	00000038	R	03
XA_IDR	00000006		
XA_INTERRUPT	000004CB	R	03
XA_ODR	00000006		
XA_READ_WRITE	0000009E	R	03
XA_REGDUMP	000005D9	R	03
XA_REGISTER	000004F5	R	03
XA_RESET_DELAY	= 00000001		
XA_SETMODE	000000F8	R	03
XA_START	0000014B	R	03
XA_TIME_OUT	00000472	R	03
XA_TIME_OUTW	0000048A	R	03
XA_WCR	00000000		

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	000000C2 (194.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	00000064 (100.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	00000636 (1590.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.04	00:00:00.61
Command processing	141	00:00:00.43	00:00:03.91
Pass 1	551	00:00:16.22	00:00:51.03
Symbol table sort	0	00:00:02.39	00:00:09.38
Pass 2	236	00:00:03.61	00:00:15.02
Symbol table output	24	00:00:00.14	00:00:00.24
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	992	00:00:22.87	00:01:20.22

The working set limit was 2100 pages.
135207 bytes (265 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2188 non-local and 58 local symbols.
1292 source lines were read in Pass 1, producing 20 object records in Pass 2.
53 pages of virtual memory were used to define 50 macros.

! Macro library statistics !

Macro library name	Macros defined
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	35
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	47

2463 GETS were required to define 47 macros.
There were no errors, warnings or information messages.
MACRO/LIS=LIS\$:XADRIVER/OBJ=OBJ\$:XADRIVER MSRC\$:XADRIVER/UPDATE=(ENH\$:XADRIVER)+EXECMLS/LIB

0117 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0118 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY